AFRL-IF-RS-TR-2002-167
Final Technical Report
July 2002

MULTI-DOMAIN NETWORK MANAGEMENT BOUNDARY DEVICE

Arca Systems, Incorporated

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 074-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503			
1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** JULY 2002	**3. REPORT TYPE AND DATES COVERED** **Final Apr 98 – Feb 02**	
4. TITLE AND SUBTITLE MULTI-DOMAIN NETWORK MANAGEMENT BOUNDARY DEVICE			**5. FUNDING NUMBERS** C - F30602-98-C-0089 PE - 63789F PR - 7820 TA - 05 WU - 01
6. AUTHOR(S) Herb Markel			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Arca Systems, Incorporated 2650 San Tomas Express Santa Clara California 95051			**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFGA 525 Brooks Road Rome New York 13441-4505			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2002-167
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Scott S. Shyne/IFGA/(315) 330-4819/ Scott.Shyne@rl.af.mil			
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			**12b. DISTRIBUTION CODE**
13. ABSTRACT *(Maximum 200 Words)* This report provides background information and conclusions for the Multi-Domain Network Management Boundary Device program. This development focused around providing a Network Common Operational Picture for a better ability to monitor diverse information enterprises composed of multiple compartmentalized networks.			
14. SUBJECT TERMS Controlled Interface, Boundary Device, Network Management, SNMP, ICMP			**15. NUMBER OF PAGES** 89 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

Figures

Tables

EXECUTIVE SUMMARY

Many military applications require merging data at several different sensitivity levels, typically coming from separate, system high networks. These networks are isolated, so that they must be individually managed. A network administrator would be able to work more efficiently with concurrent access to management data from all networks at once.

Network management protocols enable an administrator to conveniently and easily monitor and manage a network from a centralized location. Automated network management tools become vital as networks grow in scale and complexity and become host to equipment from multiple vendors. The network management protocols that are most widely used for data networks are the Simple Network Management Protocol (SNMP) and Internet Control Message Protocol (ICMP). These protocols allow for monitoring the performance of devices on the network.

AFRL and Arca Systems an Exodus Communications Co developed a boundary device to securely handle SNMP and ICMP traffic. This device, coupled with a hierarchical network management architecture, can provide a Network Common Operational Picture across multiple security domains (Secret, Unclassified, and even Coalition). This effort has significant ramifications in the operational community were there is no current capability to manage networks of varying security levels from one Common Picture. This program has successfully addressed several of the problems for coalition network management and has resulted in significant interest from allied countries including the Canada and Australia. The Multi Domain Network Management (MDNM) system provides NATO commanders with unparalleled visibility into the health and status of their information infrastructure thereby making them more effective in conducting coalition and conventional operations (refer to figure 1).

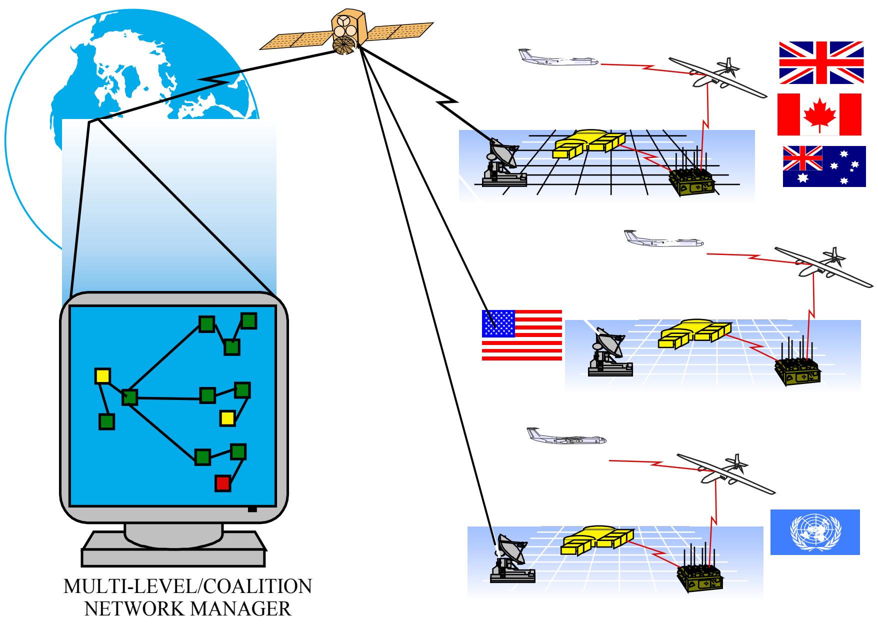

Figure 1: Conceptual Diagram

1. Introduction

Multi Level Security (MLS) military applications process data labeled at several different sensitivity levels. Often the MLS application will be trusted to merge data originating from separate, system high networks. These networks are isolated so that they must be individually managed. A network administrator would be able to work more efficiently with concurrent access to management data from all networks at once.

The Joint Forces Air Component Command is seeking robust centralized network management in such a multilevel environment. Implementation within this facility must not unduly impact the carrying capacity of component networks and should minimize the use of new mechanism development and non-standard configurations. The implementation should therefore be amenable to construction using standard COTS equipment, and the communication between network agents and the central management station must minimize nonessential message exchange. The developed solution must be easily installed, operated, and maintained.

This environment presents a number of security challenges. High to low data flow must enforce the DoD confidentiality policy: classified information must not be viewable by personnel possessing insufficient clearance. Low to high flow should enforce a credible integrity policy. Covertly embedded classified information can not be allowed in any of the data flowing from high to low. In addition, covert channels that result from modulation of the information flow must be controlled.

The development of custom proxies for common network management protocols offers a promising approach to these security issues. The network monitor will contain an engine [i.e. HP Openview's Network Node Manager], that can be controlled by the security office and utilized to determine the current state of the network information infrastructure. The engine will be such that its configuration can be set consistent with the requirements of the specific installation's certification and accreditation, but could vary from one mission to another. The information flow between the networks is controlled by the proxy server [customized firewall] which interprets specific protocols on the high side and, in response, constructs specific packets to be issued on the low side. The proxies must address potential covert information flow and not allow it to occur.

The proxies will support high to low data flows of specific formats. For instance, the SNMPv2c GetRequest, GetNextRequest, and GetBulkRequest formats could be readily supported and will provide the necessary flexibility to request needed information. The protocol fields for these commands are highly constrained, with the exception of the variable-bindings field. The variable-bindings field has specific semantics that the proxy can check. In fact, the proxy should be able to learn and store information about the component network (e.g., the actual Object IDs (OID) used) that will likely present a degree of uniformity and predictability.

This report provides the background information and conclusions for the Multi-Domain Network Management Boundary Device Program. Arca's development team refers to this program as NETMON and will use this name throughout this document.

2. Objective

The Simple Network Management Protocol (SNMP) and Internet Control Message Protocol (ICMP) are used for the remote monitoring and control of network systems from a centralized location. This development effort examines solving problems in implementing SNMP & ICMP across a classification boundary by integrating COTS technology with the smallest possible development of specialized capabilities. Security aspects go beyond the obvious concern of implementing two-way communication across a classification boundary. There now is the transmission of a vulnerability-ridden protocol across a classification boundary.

As SNMP capabilities have spread across large network environments the need for controlling the flow of SNMP messages becomes even more important. Since SNMP messages contain the basic components for taking control of any managed host the distribution and visibility of SNMP packets on a network needs to be closely monitored to prevent intentional misuse. Many enterprise networks take on the risk of passing SNMP across internal network boundaries if the gateways to external networks prevent the broadcast or receipt of SNMP traffic. Unfortunately, this still exposes systems to any internal misuse or threat.

Traditionally, firewalls are implemented over a number of logical security boundaries. Separating not only external networks but subdividing internal networks by region, work requirement, or other policy objective. Firewall vendors have implemented significant control over ICMP. Most firewall products have the granularity to control specific types, i.e. type checking for echo requests and echo responses, of ICMP packets as well as IP control. Unfortunately, firewall vendors have not responded with the same level of granularity for SNMP. Firewall products that are available and are SNMP compliant, usually mean that they will respond to an internal network management poll or will only send traps to a specified network management machine. The only other option that is available is to allow SNMP packets, with no type checking, between hosts across these security boundaries. Monitoring for SNMP traffic and validating source and destination addresses against policy maps does this. Currently, this is the extent to which firewall technology manages the flow of SNMP traffic over a network.

To address this functional deficiency, AFRL and Arca Systems an Exodus Communications Co has endeavored to develop a boundary device to securely handle SNMP and ICMP traffic. This device, coupled with a hierarchical network management architecture, can provide a Network Common Operational Picture across multiple security domains (Secret, Unclassified, and even Coalition).

3. NETMON Development Process

This section will provide background information as well as a summary to the decisions made during the development cycles of NETMON. Key steps into the development of NETMON were SNMP vulnerability assessment, firewall software/hardware identification, network management software identification, and building and testing.

3.1 Firewall Identification

3.1.1 Firewall Selection Criteria

The initial stages of this effort required examining the SNMP control capabilities of various firewalls and determining the requirements such a firewall might need to support an SNMP Proxy. This review examined six firewalls. Since the eventual goal of the firewall's implementation is to potentially sit across a classification boundary for DoD Systems more than just available proxies were considered.

Firewalls were reviewed and chosen for their operating system, ability to implement proxies on a full range of supported and unsupported protocols, ease of development, multi-level system handling, and any support of SNMP. Initially, it was intended to choose a firewall for the development effort that most closely meets the number of requirements at hand. For example, the CyberGuard firewall runs in the SCO UnixWare environment (a B1 environment) and can be certified to sit on a classification boundary. It also has a generic proxy capability that allows the execution of site specific code when packets are received on specific IP ports. (See Appendix A for firewall matrix)

3.1.2 Development Selection Criteria

After closer examination of the network architecture needed to create an SNMP Guard, a determination was made that the firewall only needs to be able to perform certain basic capabilities. These capabilities are:

- traffic flow control – to prevent all network traffic that is not SNMP traffic from a valid system
- Network Address Translation – to prevent the accidental release of 'high-side' network address data
- Split DNS – to allow each side of the proxy to maintain its own view of internal and external networks without releasing address data.

The only other main requirement to be considered was portability. How effectively could a Proxy/Guard be developed and transferred to multiple firewall platforms using the same basic code that came from the Proxy/Guard's development.

3.1.3 Firewall Conclusion

Ultimately, a simple solution was chosen for development. The FreeBSD operating system was chosen for the development process for several advantageous reasons.

Advantages include:

It's compliance and wide acceptance as a UNIX server.
Allowed access to the source code if kernal modifications were required.
Powerful IP firewall tool that allows for type control of the protocol
Base Unix operating system for three commercial firewall products. (supports portability goal)
On going project for Trusted BSD (goal is to meet B1 standards)
Arca already has security tools developed for BSD that NETMON will take advantage of.
Updates and patches are quickly available (advantage and disadvantage)

Disadvantages include:

Maintains some security features but NOT a secure OS. (May be addressed by the ongoing TBSD project)
Updates on BSD come frequently and make it hard for configuration management. (Problem with any OS)
Support of multiple Ethernet cards on a laptop was not straightforward. (Discovered in development stage.)

3.2 Network Management Software Identification

3.2.1 NMS Selection Criteria

The initial stages of this effort required examining network management software and determining the requirements such software might need to support this project. This review examined several NM software packages that were commercially available and some that were freeware.

When researching the different Network Monitoring Software, it was discovered that there were 4 basic categories of software: Monitoring & Analysis of Networks, Multipurpose or Enterprise, Developer Kits and Tools, and Application Monitoring & Analysis.

- Network Monitoring & Analysis Tools allow the system administrator to get a topographical picture and statistics of their network and computer assets. Though capable of monitoring LAN and WANs, this label is typically used to define a small business application.

- The multipurpose or Enterprise solution includes several layered products that allow the user to customize or slowly build up monitoring capabilities of large-scale networks with LAN/WAN components. Typically the Enterprise solution has the ability to monitor network assets, applications, software distribution and more.

- The developer's kits and tools allow the user to build applications that would fit into the Enterprise software.

- The Application Monitoring and Analysis Tools allow the system administrator of a distributed software environment to keep track of software distribution, performance monitoring, version control, and other aspects dealing specifically with application/software and not network asset monitoring.

Key factors were identified, after researching the different kinds of network management software and tools, to consider before choosing which software solution. These factors are:

- OS support
- Client/Server support
- Domain Management support
- Multi-Vendor support vs. proprietary solution
- SNMP Version Support
- Cost
- Expandability and Scalability
- What do you need to monitor and analyze?
- Do you want an all in one or a dedicated software program?

See Appendix B for the Network Management Software Matrix

3.2.2 Development Selection Criteria

The main development objective was to select an SNMP compliant software package that would not force the development of specialized software to support the selected network management program or lock the NETMON into that program.

3.2.3 Network Management Software Conclusion

The NETMON project built the proxy/firewall to support Hp Openview's Network Node Manager. This selection was based on several factors.

Advantages include:

Availability and wide use across the U.S. Air Force.
HP Openview also supports the key factors mention above.
Openview supports multiple UNIX operating systems as well as NT.
It supports the Client/Server environments.
Allows hierarchical domain management support.
Expandable and scaleable.
Demo version available for use.
Training and Support available.

Disadvantages:

Cost for a full license is prohibitive for a small project
Cost of support contract is prohibitive for a small project.
Proprietary application of features discovered during development of NETMON.

HP Openview was determined to be the network management software of choice to readily meet user requirements. The obstacles of using HP Openview were alleviated with the Government providing a fully licensed and supported version for the development and demonstration facilities.

Note: To fulfill the objective of not building a solution that was dependent on a particular network management solution, ARCA did use other available tools such as Scotty and TKIned, to test overall compatibility with other products as time permitted.

3.3 SNMP Vulnerability Assessment

This section will provide background information on the SNMP protocol and a summary of the vulnerabilities discovered. Refer to the Network Management Protocol Vulnerability paper, also entitled Protocol Select/Analysis White Paper, CDRL A003, Contract # F30602-98-C-0089 for the complete assessment.

3.3.1 SNMP History

The Simple Network Management Protocol (SNMP) was developed in 1988, and has become the de facto standard for internetwork management on TCP/IP networks. Besides being a standard, a wealth of compliant products are available. SNMP is referred to as “simple” because the associated software on remote network components is minimal. Most of the processing power and data storage resides on the management system. This simplifies the design of network components that support the protocol and has promoted its widespread acceptance.
Towards the goal of simplicity, SNMP supports a limited set of management commands and responses. SNMP allows managers (the console station a human network administrator uses) and agents (the entities that interface to the actual devices being managed) to communicate for the purpose of accessing managed objects. SNMP can be used to find out how much traffic flows through various router interfaces and look for network congestion, or to monitor host-specific information on other devices. Hosts, bridges, communication servers, printers, routers, hubs, etc. can be equipped with SNMP agents. There are no specific guidelines in the standards as to the number of management stations, or the ratio of the number of management stations to the number of SNMP agents. Relatively few management stations are required with many SNMP agents due to the “simpleness” of the protocol.

SNMP communication involves reading the values of the objects within a virtual database and altering the values as appropriate to monitor and control devices on a network. The managed objects may be hardware, configuration parameters, performance statistics, etc. The database is a virtual information database called a Management Information Base (MIB). The MIB functions as a collection of access points at the agent for the management station. It may be depicted as a tree when the leaves are the individual data items. Vendors can define their own private branches of the MIB tree.

The security vulnerabilities introduced by unnecessary or misconfigured SNMP are often not well addressed in security audits. Administrators will often leave routers, modems, hubs, bridges, hosts, and other devices including firewalls with readable and writeable communities enabled. These allow remote users (or possibly a determined hacker who is an outsider) to query devices on local networks and discover a staggering amount of information about local systems and network layouts. Stealthy requests via SNMP allow attackers to mine sensitive configuration data to facilitate a subsequent attack. A hacker who obtains physical access to a network (e.g., receptionist desk) can use a protocol analyzer to develop a thorough blueprint of network topology and configuration. Access control is weak because it is based essentially on a shared secret.

At the time of this research, the primary fielded versions were SNMPv1 and SNMPv2c. There were very few fielded SNMPv2 products that had compliance with the stated specification. Mostly what was found was the SNMPv2c, which is essentially v1 with some new commands. Finally, SNMPv3 had only recently been released. A number of companies had promised the release of SNMPv3 products, but taking a you-go-first approach.

Ultimately the development team of NETMON decided to only handle the SNMPv1 and SNMPv2c for the prototype. This decision was driven with the knowledge that HP Openview and the other network management software that was freely available to Arca, Scotty and TKIned, only SNMPv1 and SNMPv2C.

3.3.2 SNMPv1 Description

The SNMPv1 is a simple query/response protocol that has five simple types of messages called PDUs (Protocol Data Units). A PDU can contain all or part of a message exchange.

Five PDU types are:

1. **get-request** - Request to retrieve a specific MIB system variable.
2. **get-next-request** - Request to retrieve next variable in a MIB table.
3. **get-response** - Response to a get-request or get-next-request PDU.
4. **set-request** - Request to alter a MIB variable (a system setting).
5. **trap** - Asynchronous notification of an event (agent to manager).

Remote administrators can use the **get-request** message to retrieve information from remote sites, routers, and the other devices that have an SNMP agent. The agent responds with a **get-response** message and may return such information as the name of the system, how long the system has been running, and the number of network interfaces on the system. The **get-next-request** is used to walk down a table (e.g., an IP routing table) in lexigraphical order. An administrator uses the **set-request** message to manipulate a variety of settings on a target device. This PDU can be used to set the name of the device, change an interface address, shut down an interface, or clear an address resolution table entry. Obviously it is useful to be able to remotely find configuration information and remotely manage devices without physically visiting a site and pulling a unit out. Attackers also find this powerful configuring capability attractive to exploit. The **trap** message is an unsolicited message an agent sends to a station to inform it about the occurrence of a specific event (e.g., a user has logged into a host, disk space is nearing capacity, authentication failure, etc.). Enterprise-specific traps (defined by vendor) greatly outnumber generic traps and can be triggered based on wide variety of events.

SNMP (ISO layer 7) uses UDP (User Datagram Protocol) as the transport layer (ISO layer 4) protocol mechanism, and Internet Protocol as the network layer (ISO layer 3). The basic layouts of SNMPv1 packets are:

Table 1: Basic Layouts of SNMPvl packets

get-request, get-next-request, set-request

Version	Community	PDU type	Request ID	0	0	Name X	Value X	- - -

get-response

Version	Community	PDU type	Request ID	Error Status	Error Index	Name X	Value X	- - -

trap

Version	Community	PDU type	Enterprise	Agent Address	Generic Trap	Specific Trap	Time	Name X	Value X

Community is SNMPv1's authentication mechanism. PDU type designates which of the above 5 types of messages are being sent. Request ID is used to differentiate between requests. Error status returns error messages, and error index gives the offset of the variable which was in error. Finally, name and value represent the name of the field requested and either the value to set it to or the value of it on the remote server. These are defined by a MIB written in ISO Abstract Syntax Notation One (ISO ASN.1), and encoded using a code called Basic Encoding Rules (BER). ASN.1 is used to define data and the types and properties of this data. BER is used to actually transmit the data in a platform independent manner. The values that can be fetched and set via SNMP are defined in what is called the Management Information Base (MIB). The MIB is written in ASN.1, and defines all the different variable classes, types, and variables associated with SNMP. Information stored in the MIB is done in classes. Classes are used to define variables associated with data for statistics and values for the system as a whole. Other information that may be obtainable, depending on the system agent, includes interfaces on the system, an address translation table, and protocol (IP, TCP, UDP& ICMP) statistics. The information and structure of a MIB often will vary depending upon originating vendor. Variables may hold similar information yet be labeled and stored in differing manner.

Drawbacks affecting SNMPv1 are that it is officially standardized only for use on IP networks and it is inefficient for large table retrievals. SNMPv1 does not support a standardized way to manage SNMP agents themselves via SNMP and must contend with non-standardized, often redundant MIB extensions / datatypes from hundreds of vendors. SNMPv1 lacks an effective security model, and in the following section we note the substantial security vulnerabilities of this protocol, headed by the transmittal of cleartext strings for authentication.

3.3.3 SNMPv2 Description

The Simple Network Management Protocol version 2 has the same basic functionality as SNMPv1: to acquire and change MIB data on network devices. This version is quite confusing to discuss since the version that was defined as the reference is rarely seen in the field. The majority of network management today still relies on the original SNMPv1 specification, with limited numbers of products running something of v2 flavor. The focus of this effort was on the SNMPv2c flavor.

The "compromise" version SNMPv2c is the most common variant seen in the field (e.g., HP's Network Mode Manager). SNMPv2c is the combination of the enhanced protocol features of SNMPv2 without the SNMPv2 security. The "c" comes from the fact that SNMPv2c defaulted back to the SNMPv1 community string paradigm for "security".

The SNMPv2 protocol has seven PDU types:

1. **get-request** - Request to retrieve a specific MIB variable.
2. **get-next-request** - Request to retrieve next variable in a MIB table.
3. **GetBulkRequest** - Request to retrieve an entire MIB table.
4. **response** - Response generated by any other PDU.
5. **set-request** - Request to alter a MIB variable.
6. **snmpV2trap** - Asynchronous notification of an event (agent to manager).
7. **InformRequest** - Asynchronous notification of an event (manager to manager).

(Note that SNMPv1 PDU's adopted a hyphen between name fragments, and SNMPv2 does not.)
The protocol messages in SNMPv2 are very similar to those in SNMPv1. Both protocols have get-request, get-next-request, set-request, get-response, and trap messages. The SNMPv2 GetBulkRequest can be seen as a series of SNMPv1 get-next-request messages. These similarities simplify the implementation of a bilingual manager that speaks both of the protocols (and which will still house the security vulnerabilities discussed earlier.) The basic layouts of SNMPv2 packets are:

Table 2: Basic Layouts of SNMPv2 Packets

get-request, get-next-request, set-request, snmpV2trap, InformRequest								
Version	Community	PDU type	Request ID	0	0	Name X	Value X	- - -

response								
Version	Community	PDU type	Request ID	Error Status	Error Index	Name X	Value X	- - -

GetBulkRequest								
Version	Community	PDU type	Request ID	Non-repeaters	Max Repetitions	Name X	Value X	- - -

While SNMPv1was originally defined for transmission over UDP and IP, SNMPv2 pursued expanded transport mappings. These included OSI Connectionless-mode Transport Service (CLTS), AppleTalk Datagram Delivery Protocol (DDP), and Novell's Internetwork Packet Exchange (IPX). SNMPv2 over UDP is the preferred transport mapping to maintain compatibility with SNMPv1, which therefore results in the same security issue as SNMPv1.

3.3.4 SNMPv1 and SNMPv2c Security

The majority of vulnerabilities in SNMPv1 and SNMPv2c, referred to as SNMP for the rest of this section, are at the design level and cannot be "corrected" without a wholesale transfer to other network management protocols. The most disturbing vulnerability is that SNMP uses cleartext community strings for security. An attacker who can capture packets off a network segment can capture the community string. Even if the attacker can't sniff the network segment, default community names are often left active and are not shy about returning useful information to assist the brute forcing of legitimate private community names. With a valid community name, the attacker will be able to change configuration of the associated network devices that use SNMP and allow SNMP **set-request** messages.

3.3.4.1 Underlying SNMP Mechanisms Are Connectionless

SNMP is an Application Layer protocol that relies on protocols at the lower OSI layers for other communication functions. It assumes that its underlying communication path is a connectionless communication subnetwork. SNMP utilizes the User Datagram Protocol (UDP) to transport it through the internetwork, and the Internet Protocol (IP) to provide Network Layer functions, such as addressing. Together these protocols comprise an IP datagram:

Table 3: IP Diagram

SNMP Message Within a Transmission Frame				
Local Network Header	IP Header	UDP Header	**SNMP Message**	Local Network Trailer

←UDP Datagram→
←IP Datagram→
←Local Network Frame→

UDP (as well as IP) are connectionless protocols providing host-to-host transmission of data without first establishing the communication channel. UDP does not provide a reliable method for exchanging messages between an agent and a station. Because SNMP relies on UDP, SNMP is itself connectionless. No ongoing connections are maintained between the management stations and respective agents. SNMP makes no guarantees about the reliable delivery of data, and datagrams that do not get to their destination must be retransmitted.

A connectionless protocol is problematic in any application requiring security. With a connectionless transport, an attacker could reorder, delay, or copy and later replay a SNMP message. An example of the latter would be an attacker's copying and then replaying a message to reboot a device at an inopportune time. A connectionless protocol also simplifies attacker-spoofing efforts. An attacker can easily spoof packets to a server, and modify/add/reconfigure the state of the server. By using a spoofed source address, there isn't any way to get the return message. The machine being spoofed will simply drop the response message, and the server is none the wiser. Using the 'snmpset' program modified to use a raw socket to allow source address forging, an attacker can modify any value in the MIB defined as read-write as long as he has knowledge of a privilege community name.

3.3.4.2 Cleartext Community Names

SNMPv1 can authenticate based on two different methods: source address and the use of "community" names. Current SNMP agents use IP address-based access lists. However, the source address is trivial to forge, rendering this approach useless against directed attacks.

Community names can be thought of as passwords to the SNMP agent -- the sender of a request must know the remote community name, which is configured in the /etc/snmp.conf file. Each SNMP request includes the remote community name to "authenticate" the access request to the snmpd program on the target. This authentication is extremely weak. SNMPv1's deficient security implementation can probably be attributed to the premium placed on impact to CPU and memory resources a decade ago. Cleartext community names were inexpensive to implement against infrastructure device firmware (and the cyber-world was a less hostile place back then).

Readily available and inexpensive packet sniffers and protocol analyzers make it reasonable to conclude that community names sent in the clear are a fatal security deficiency. Every SNMP packet offers the community name in cleartext. SNMP uses port 161. The target community name field is typically 6-8 characters long. If an attacker is able to apply a sniffer (e.g., *snoop*) to an active network segment it is easy to capture the password.

If an attacker is successful in capturing the cleartext community name, he has the ability to request information and modify the respective parameters.

3.3.4.3 Default Community Names

If an attacker cannot gain access to sniff the segment for cleartext community names, he may still be able to collect pertinent site information via default community names. The default 'public' community is frequently left undisturbed because administrators, not realizing the risk it poses, do not take the time to deactivate this community. However, attacker information requests against this community often provide enough data to permit brute forcing a legitimate sensitive community name.

Similar to guessing passwords, one can use this information to try to piece together a community name. Popular favorites include stuff like 'admin' 'router' 'gateway' and the like, combined with numbers or guessable characters. While failed attempts are noted, very few administrators, if any, ever check for them. There is also no timeout or locking of connections without add-on mechanisms. In face of the possibility of brute guessing of passwords, it is important that use of strong passwords be enforced. The rules of thumb mandated for user authentication apply here as well. Do not use words found in dictionaries or simple variations thereof. Do not use locally meaningful (and guessable) words. Use non-dictionary names with a suitable random mix of numbers and other non-alpha characters.

Even if only public works, there are lots of interesting things available via SNMP. One can dump routing tables, connection tables, statistics on router use. In certain situations, one can even get information on packet filters in place, and access control rules. All are useful information to have in setting up attacks in conventional manners. Sometimes public is even given r/w on certain tables, and one can do most of what they need to do via that account.

3.3.4.4 Further Snooping for Network Configuration Data

The following are scenarios for collecting further network configuration data via SNMP:

1. A remote user wants to gain access to a network, but wants to break into a machine that would offer the optimal benefit for the effort. For example, a machine located on a network that is used for connecting routers, and allows the most profitable password sniffing would be targeted. By making a few SNMP queries to routers, the network topology can be easily mapped out, and attacks can be concentrated to the most worthy target machines.
2. A user on the local network is running an NT fileserver, with default SNMP enabled. By sending a query to the host a user can discover the "real" NT system name, used in file sharing. (It does not need to be the hostname, and frequently is not. Without this name, you are not able to get filesharing access.) By using a remote PC (or samba on a UNIX machine), the server can easily be scanned to check for open fileshares. A quick password guessing attack can also be performed.
3. A remote user can iteratively scan remote addresses, looking for SNMP manageable devices. Scanning a network and asking each host for its system.sysDescr.0 (system description) will produce a nice list of each system and OS on the network that is manageable. Many PCs, Macs, UNIX systems, Xterminals, modems, terminal servers, printers, print servers, hubs, switches, and routers all have a default public community and allow easy browsing.

The above situations have been more passive and mainly information gathering, but a denial of service attack is quite reasonable. If a writeable community string is found, the system settings may be modified. Network interfaces can be shut down, modems disconnected, and routing tables modified.
One interesting attack could be used in conjunction with network spoofing. If the interface on a hub or router for a machine could be shut down, spoofed packets from that machine be sent, and the interface restored, blocking unwanted responses from that machine is easily taken care of. (This would be in the TCP sequencing attack, where the spoofed host is flooded with opening connections to prevent a response at an inopportune time.)

3.3.4.5 Remote Manipulation

If an attacker gleans a community name, and uses the ability to forge the manager (the SNMP client) address, he can remotely configure network devices. All the attacker needs to figure out is what is useful to modify. This can vary. There are sets of default configuration values that almost every SNMP'able machine will have. In addition to these, though, are the 'enterprise' MIB's, which define vendor specific SNMP tables and fields (e.g., specific to Cisco, Ascend, etc. and per device). Definition of these MIB structures is easily found.

3.4 MLS Network Management Issues

Utilizing SNMP in an MLS environment introduces concerns about disclosure of high data. Any facility that permits information flow from a high agent to a low station, or a high station to a low agent, may introduce an overt or covert information channel. This section provides background about current covert channel theory and practice, and discusses covert channels within the context of MLS network management.

3.4.1 Overt and Covert Channels

A *covert channel* is a means by which subjects can communicate contrary to the organization's Mandatory Access Control (MAC) policy, through the exploitation of information channels that the system designers did not intend to be used for such communication.[1] In somewhat more detail, a covert channel is any means by

[1] It is important to note that often, a covert channel operates in full compliance with the *system's* access control policy, even though it is contrary to the *organization's* access control policy! This is because the formal statement of the system's policy has either been formulated to support an exploitable interlevel communication mechanism or because of an oversight or omission on the part of the designers.

which one untrusted process, the covert channel *sender* (*S*), can communicate information to a second untrusted process, the *receiver* (*R*), where the sensitivity level of *R* does not dominate that of *S*.
On a properly architected MLS system, MAC controls all means of interprocess communication (IPC) intended to transfer data. Where this is not the case, the resulting information paths are sometimes called *overt* channels. Overt channels are just those that can readily be eliminated because they can be directly controlled by MAC. A channel that only occurs when the level of *R* dominates that of *S* does not violate the basic MAC policy, and is therefore not termed a 'covert channel'.[2]
Covert information channels are a means by which to signal information via computer system facilities not intended for authorized data transfer under the information sharing policy. Actual use of this signaling mechanism may or may not be intentional on the part of both participants. Indeed, the information transfer may be effectuated in ways not normally detected or regulated by the access control mechanism of the Trusted Computing Base. The following are two informal definitions of covert channels:

- A channel exists when a subject R in violation of the organization's security policy can detect an action by some subject S.
- A channel exists when a subject *S interferes with* subject *R* such that inputs to *S* affect outputs from *R* in violation of the organization's security policy.

Covert channels are *necessarily* inherent in multilevel systems. It is difficult to eliminate completely all perception of the controlled sharing of common resources[3] across access-class boundaries on an integrated computing platform. One of the primary objectives of a multilevel system is to share labeled data in conformance to defined MAC policy. The presence of covert channels in such systems should not come as a surprise. Every bit of information in a system where modification done by one process can be read by another process, whether directly or indirectly, is potentially a channel.
The search for covert channels is straightforward in theory, though it has thus far proved generally difficult and tedious in practice. Heavy dependence typically must be placed on the analyst's skill, intuition, and deep understanding of the target system and its underlying logic. Analysts must assess each resource in the system (which could be as small as a bit or as transient as the position of a hard disk arm or the availability of a buffer in a network protocol implementation) and determine under what conditions an untrusted process *S* can alter its state and under what conditions another untrusted process *R* can detect alterations in *S*'s state or the *fact that such alterations have taken place*. If it could ever be the case that *R* can detect an alteration by *S*, and that at the same time the sensitivity level of *R* does not dominate that of *S*, then a covert channel exists. The analysts must then make an assessment of the channel bandwidth to determine the practicality of exploiting the channel. The effort needed for closure of an identified channel typically falls into the category of either *easy* (obvious and likely exorcised in the design process), *costly* (recognizable, but painful in terms of performance or functionality to close), or *impracticable* (the effort to close is intractable).
The trade-off between providing essential system properties and the confinement of covert channels is often difficult. While covert channel analysts seek out channels and estimate their bandwidth (and may occasionally produce suggestions for confinement), implementing covert channel confinement must be an engineering judgment coordinated with competing engineering goals and organizational development priorities.
The theory behind covert channel analysis has been fairly mature for decades. Finding ways to apply theory to permit practical, efficient detection and arresting of covert channels has seen slow progress, and remains more as an art than a science. There have been limited advances in recent years, particularly in attempts to address the difficult problem of detecting insidious timing channels. There is cause for hope that renewed focus on covert channel research will yield promising approaches and additional beneficial results.[4]

---

[2] It may be a *direct storage channel* or a discretionary Trojan horse.

[3] This is true even for those common resources that are *partitioned* by partially-ordered sensitivity levels.

[4] One possible explanation for limited progress in covert channel research is the impression held by many that there are enough windows open in most operating systems for a penetrator to crawl through and simply steal the chestnuts, that worrying about smoke signals wafting out the chimney from their roasting has understandably taken lower priority. As the vulnerabilities in modern secure systems are reduced, the need for better, more systematic solutions to exploitable covert channels increases in importance.

3.4.2 Timing vs. Storage Channels

A *timing* channel is one in which *R* receives information only through measurement of differences in the passage of time. In a timing channel, *S* can affect the duration or timing of some operation that *R* can perform. *R* measures the time consumed by its operations, or the intervals at which certain events take place, and from those values, and those values only, determines the covertly signaled information.
All other channels are *storage* channels (sometimes called "resource" channels). Storage channels always involve some entity (the "resource") with a state that *S* can *modify* such that *R* can *detect* the modification. The use of time for synchronization of channels does not in itself classify them as timing channels. Storage channels may use time-critical events to synchronize their efforts. For example, *S* might change the state of the resource at regular intervals and *R* could query the resource at the middle of each interval.
It should be recognized that there is a continuum between "pure" covert storage channels and "pure" covert timing channels. Some channels cannot be clearly identified as either storage or timing channels, but have aspects of both. Further, most channels can be modified in implementation to be more "purely" storage- or timing-dependent in operation. One practical consideration in differentiating a channel as being either storage- or timing-dependent is that the DoD TCSEC requires that covert storage channels be dealt with at the B2 level, while the B3 level criteria requires both storage *and* timing channels be addressed.

3.4.3 SNMP and Covert Channels

There are several common sources of covert channels. The detailed attributes, as distinguished from the contents, of MAC-controlled data objects often escape MAC control. Internal OS kernel variables can be serially shared between system calls at different MAC levels without being reinitialized[5]. The ability to detect the existence or non-existence of resources at a non-dominated level can also be a channel.
Perhaps the largest source of channels involves the allocation of resources to subjects at different levels from the same global pool of potentially shared resources. When both *R* and *S* allocate from the same resource pool, there are many potential channels.
Both timing and storage channels must be considered at the boundary line where network management is crossing security levels. These must be eliminated or reduced as much as possible. If the low side can see the messages as they cross from the high, then the high messages must be ensured to not signal covertly high data. The use of SNMP proxying for the protected enclave can be used to interrupt potential channels by request batching, protocol header field blanking, implementing delay insertion, etc.
Classic techniques may be used to address identified covert storage and timing channels. Requests for specific information may be supplemented by additional requests, their order changed and variable timing delays inserted. Multiple GetRequests and GetNextRequests could be canonically ordered or shuffled and batched into a single GetBulkRequest PDU. The Proxy could make audit trail entries for detected events indicating potential exploitation of a known covert channel. The degree of channel throttling could be made site selectable.

3.5 Design and Build

The next task was to design a prototype and architecture that could meet the objectives of creating a firewall module or implementation that handles network management protocol traffic to be controlled across classification boundaries and mitigating the security concerns discovered from the research discussed above.

The first task in undertaking the design was the creation of the following requirements/goal list:

- Commercial Off-The-Shelf products (COTS), minimal Government Off-The-Shelf (GOTS)
  - Hardware, Firewall software COTS
  - Developed software the only GOTS
- Built to interface with HP Openview's Network Node Manager
- Built to be independent of HP Openview (i.e. no special HP Openview module)
- Proxy Network Management protocols
  - SNMPv1,

[5] Technically these cases can often be considered a flaw in object reuse.

    - SNMP v2c,
    - ICMP[6]
- Configurable SNMP Security Policy
- Format validation checks on all packets being proxied
- Content filtering on all network management packets being proxied by the BD
- Block all other packets from entering system (firewall rules)
- Does NOT directly route network traffic but instead proxy all packets (generate new packet of similar or identical content)
- Boundary Device must reside in a protected environment
- Design to be automatic (NO MAN IN THE MIDDLE needed for release)
- Handle and Minimize covert channels

The following sections break down the derived requirements and addresses how each requirement was met.

3.5.1 COTS products, minimal GOTS

3.5.1.1 Hardware, Firewall software COTS

During the course of the program the development effort took advantage of using several laptops (to keep the footprint small) and desktops to operate NETMON. The following is are the minimal hardware and software requirements derived from the experience.

Minimum Hardware requirements:

- Pentium class computer capable of running FreeBSD 3.1 or higher. The system BIOS should preferably support booting from devices such as CDROM and or SCSI.
- 64MB – 128MB RAM
- 1, 1GB – 4GB HD
- 1, 1.44MB Floppy
- 1, 24X or faster CDROM (IDE)
- 2, 10/100Mbps Ethernet Cards
- 1, mouse
- 1, SVGA or higher graphics adapter
- 1, 15” – 17” multisync monitor that can support 1024x768 @ 75Hz
- 1, 30 – 60 minute UPS (optional)
- RD/RW CDROM (optional)

Minimum Software requirements:

- FreeBSD 3.2
- IP Firewall (comes with FreeBSD)

[6] This requirement came after the first prototype was built and it was discovered HP Openview required the ability to accomplish an ICMP: echo request (known as ping) and a netmask address request.

3.5.1.2 GOTS

The only government product from this development is the actual code developed that handles the proxying, screening, context checking, and user interface. The eventual goal has been that this product will be transitioned to a commercial entity that would productize this portion in part or whole.

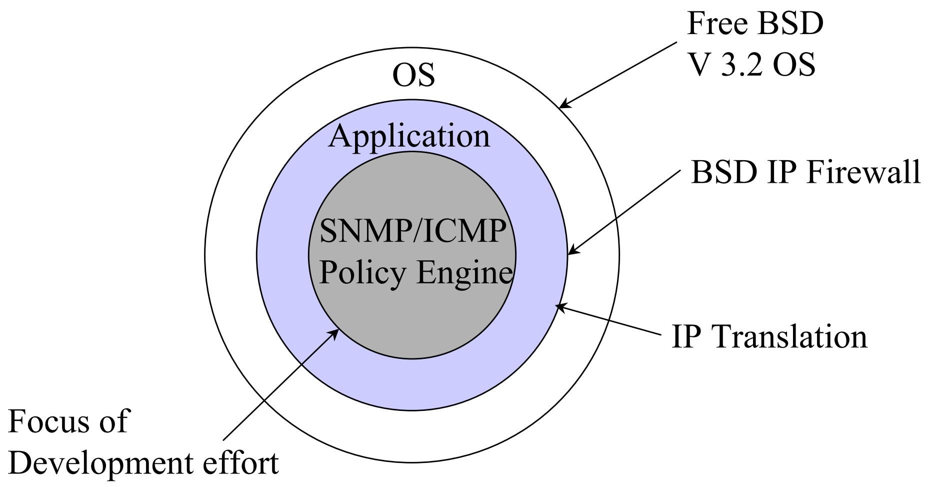

Figure 2: Software Architecture Concept

3.5.2 Built to interface with HP Openview's Network Node Manager

HP Openview was provided by RL for integration of the system. HP Openview was configured to support Hierarchical management. Each domain is still required to have a Network Management Station (NMS). An NMS is used to collect and display information about the nodes (computers, routers, bridges etc. A secondary function of the NMS is to report the information to an assigned Master or Centralized NMS. The Centralized NMS collects information from multiple collection stations and then correlates the information and displays a combined operational picture. See figure 3 for a typical hierarchical network management architecture.

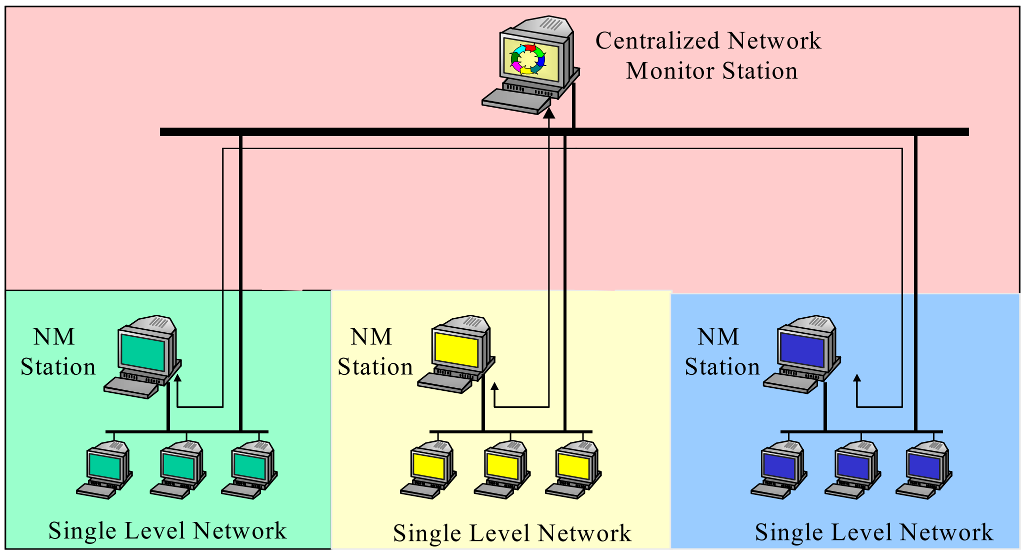

Figure 3: Typical Hierarchical Management Architecture

3.5.3 Built to be independent of HP Openview (i.e. no special HP Openview module)

Another goal for this effort has been NOT to make the NETMON device dependant on the selected network management software selection. This included two aspects. The first is that any network management software solution that uses SNMPv1 or SNMPv2c should be able to work through this device. The development team has tested NETMON with other network management programs such as Scotty and Tkined. These programs would act only as a NMS and did not support the hierarchical management scheme. Therefore, the only fact that can be stated is that the NMS could communicate with a node and receive a response through the device.

The second aspect to this goal was to NOT have to develop any special module that would need to be added to the network management software, i.e. HP Openview module, or to the boundary device. It was desired that if any code needed to be included into the NETMON device itself that it could be coded in such a way that it would work automatically when needed. It was not desirable to have to select a 'HP Module' or a 'Scotty' module when compiling the NETMON software.

3.5.4 Proxy Network Management protocols

3.5.4.1 SNMPv1

As described before, the SNMPv1 is a simple query/response protocol that has five simple types of messages called PDUs (Protocol Data Units). A PDU can contain all or part of a message exchange.

Five PDU type are:

1. **get-request** - Request to retrieve a specific MIB system variable.
2. **get-next-request** - Request to retrieve next variable in a MIB table.
3. **get-response** - Response to a get-request or get-next-request PDU.
4. **set-request** - Request to alter a MIB variable (a system setting).
5. **trap** - Asynchronous notification of an event (agent to manager).

The proxy must be capable of distinguishing the above 5 PDU types and appropriately associate them with the SNMPv1 definition.

3.5.4.2 SNMP v2c

The Simple Network Management Protocol version 2c has the same basic functionality as SNMPv1: to acquire and change MIB data on network devices. The major changes was the addition of the GetBulkRequest and InformRequest PDU, and changes to the trap format.

The SNMPv2 protocol has seven PDU types:

1. **get-request** - Request to retrieve a specific MIB variable.
2. **get-next-request** - Request to retrieve next variable in a MIB table.
3. **GetBulkRequest** - Request to retrieve an entire MIB table.
4. **response** - Response generated by any other PDU.
5. **set-request** - Request to alter a MIB variable.
6. **snmpV2trap** - Asynchronous notification of an event (agent to manager).
7. **InformRequest** - Asynchronous notification of an event (manager to manager).

The proxy must be capable of distinguishing the above 7 PDU types and appropriately associate them with the SNMPv2c definition.

3.5.4.3 ICMP

The original plans did not include for the handling of the ICMP protocol. This requirement came after the first prototype was built and it was discovered HP Openview (and others programs) required the ability to accomplish certain ICMP commands. ICMP has several commands but there are only 4 that are required to be handled.[7]

echo request (known as ping)
echo response
netmask address request
netmask address response

3.5.5 Configurable Security Policy

The security policy must be implemented in such a way that full control of each command, SNMP protocol (v1 or v2c), ICMP, and direction (flow control) can be accomplished.

The following table reflects the default and recommended settings for the SNMP security policy, which is set in a configuration file. The recommended ICMP Security Policy Settings, Table 2, is currently controlled though the firewall configuration file and not in the developed code.

Table 4: Recommended SNMP Security Policy Settings

COMMAND Or TYPE	DESCRIPTION	HIGH TO LOW FLOW CONTROL	LOW TO HIGH FLOW CONTROL
Set	Assigns a value to a variable	Conditionally Permitted	Not Permitted
Get	Returns value of Variable(s)	Permitted	Not Permitted
GetNext	Returns value of list of Variables	Permitted	Not Permitted
GetBulk	Returns a Number of Variables	Permitted	Not Permitted
Inform	Transmits unsolicited Information between NMSs	Not Permitted	Not Permitted
TRAP	Transmits unsolicited Information from managed object to NMS	Not Permitted	Permitted
Responses	Responses to commands	Not Permitted	Permitted

Conditionally in this case is based on the content screening of the data fields. Certain fields may have to be changed to protect high side information being transmitted down.

[7] The IP firewall software is used for type checking for the ICMP commands. The current implementation in the proxy code is a stub routine. In other words no parsing of the ICMP data is performed. Consequently, the return value is always pass. Further discussion will follow in the Future Enhancement section.

Table 5: Recommended ICMP Security Policy Settings

COMMAND Or TYPE	DESCRIPTION	HIGH TO LOW FLOW CONTROL	LOW TO HIGH FLOW CONTROL
Echo Request	Ping with an IP number	Conditionally Permitted	Not Permitted
Echo Response	Return for ping	Not Permitted	Permitted
Netmask Request	Requests the Netmask address for a target	Conditionally Permitted	Not Permitted
Netmask Response	Returns the target's Netmask address	Not Permitted	Permitted

Conditionally in the ICMP type is based on source and destination. See firewall rules 3.5.8

3.5.6 Format validation checks on all packets being proxied

When the boundary device is in operation all UDP SNMP and certain types if ICMP packets are allowed into the firewall and subsequently diverted to the proxy daemons that are listening on two distinct ports. At present and by default high2low and low2high listen on ports 9696 and 6969 respectively (see firewall rules 3.5.8). There is nothing special about the choice of ports, and both proxy daemons are designed to run on any divert port that has been specified in the configuration file. Divert sockets are a required component of the boundary device. Without them, datagrams would never reach either of the proxy daemons.The 'divert' captures all the packets that are allowed to come into the firewall and sends those packets to the respective proxy daemon. The proxy daemon is responsible to ensure that the packets are either type UDP SNMP or are of type ICMP.

If type UDP SNMP: Source and destination are checked against the security policy (configuration file). Next the packets are parsed and the format is verified against the SNMP protocol specification (RFP) and against the SNMP security policy. If at any point the format is wrong, the PDU type is not permitted, or source and destination check fail, the packets are disposed off and not proxied.

If type ICMP: Source and destination are checked against the security policy (configuration file). Next the packets would normally be parsed and the format verified against the ICMP protocol. If at any point the format is wrong, the PDU type is not permitted, or source and destination check fail the packets are disposed off and not proxied.

The current parsing implementation is a stub routine only. In other words, no parsing of the ICMP data is performed. Consequently, the return value is always pass. During this effort the flow control (security policy) and type checking of the ICMP commands has been accomplished at the firewall application layer. The stub process was inserted to allow for the future enhancement of accomplishing the same filtering on this protocol.

3.5.7 Content filtering on all network management packets being proxied

This function is also contained within the proxy daemons. As the packets are being checked for format, the fields are also being checked for content. For example if the PDU type 9 shows up the packet would be rejected since there are only 7 types. Another example of the checking is that if a PDU type 6 shows up with a SNMP version of 1 the file would be rejected for there is no type 6 in SNMPv1.

There is also an added filtering not based on the RFP. The search is specifically for the hostname and IP address of the high side central NMS. It came to the development teams attention that HP Openview, when configured for hierarchical network management, passed down the hostname and IP address during the initial setup of the collection station (low side NMS). This information was telling the collection station where and who to send the db information to. The proxy daemons therefore have to search the data structure for this information and substitute it with the low side information of the firewall. Upon receipt of the new proxied packet the collection station now thinks the firewall is the central or master NMS.

Again, the above paragraphs in this section only refer to the SNMP portion of this effort. The ICMP is strictly handled by the firewall rules.

3.5.8 Block all other packets from entering system (firewall rules)

Through the use of the IP firewall rules, the only UDP and ICMP TYPE 0,7,17,18 are allowed to enter the system. Below is the table of firewall rules currently in use with a detailed explanation of what each line represents and accomplishes.

Table 6: Example IP Firewall rules for NETMON

```
00100 allow ip from any to any via lo0
00200 deny ip from any to 127.0.0.0/8
00300 divert 6969 udp from 192.168.4.131 to 192.168.4.200 via ep0
00400 divert 9696 udp from 175.75.75.5 to 192.168.4.131 via ep1
00500 divert 6969 icmp from 192.168.4.131 to 192.168.4.200 via ep0 icmptype 0,18
00600 divert 9696 icmp from 175.75.75.5 to 192.168.4.131 via ep1 icmptype 8,17
00700 allow icmp from 175.75.75.1 to 175.75.75.5 via ep1 icmptype 0,18
00800 allow icmp from 175.75.75.5 to 175.75.75.1 via ep1 icmptype 8,17
65000 deny log ip from any to any
65535 deny ip from any to any
```

There are two firewall rules needed for normal operation of the boundary device. These rules are as follows:

```
ipfw add 100 pass all from any to any via lo0
ipfw add 200 deny all from any to 127.0.0.0/8
```

Rule 100 exists to allow local loopback traffic to pass between client and server applications that run on the boundary device. As the boundary device becomes more of a specialized device, it may be possible to remove these rules. For now it is not known what other parts of the system would cease to operate if traffic on the loopback interface were to be blocked by the absence of this rule. Remember, the final rule for IPFW is always deny everything that isn't specifically allowed.

Rule 200 is a protection rule. Its function is to block spoofed IP packets arriving on any interface.
In order for low2high to actually receive packets, the IPFW facility must be installed and configured with rules that are similar to the following:

```
ipfw add 300 divert 6969 udp from <lowNMSip> to <lowBDip> via <lowif>
ipfw add 500 divert 6969 icmp from <lowNMSip> to <lowBDip> via <lowif> type 0,18
```

The icmp type, numbers have the following meaning:

icmp type 0:	echo response
icmp type 18:	netmask address response
icmp type all others:	not allowed

In order for high2low to actually receive packets, the IPFW facility must be installed and configured with rules that are similar to the following:

```
ipfw add 400 divert 9696 udp from <highNMSip> to <lowNMSip> via <highif>
```

```
  ipfw add 600 divert 9696 icmp from <highNMSip> to <lowNMSip> via <highif> type 8,17
```

The icmp type numbers have the following meaning:

```
        icmp type 8: echo request (ping)
        icmp type 17: netmask address request
        icmp type all others: not allowed
```

The next 2 firewall rules are to allow the high side network manager to be able to "see" the high side of the boundary device. The firewall will NOT respond with an SNMP request. Therefore, the network management software will only identify the BD as a box on its network. The low side of the BD will not respond to a ping or netmask address request.

```
ipfw add 700 allow icmp from < highBDip > to < highNMSip > via < highif > type 0,18
ipfw add 800 allow icmp from <highNMSip> to <highBDip> via <highif> type 8,17
```

The icmp type numbers have the following meaning:

```
        icmp type 8: echo request (ping)
        icmp type 17: netmask address request
        icmp type all others: not allowed
```

It is possible to allow the low side network manager to "see" the low side of the BD only. The information would then be passed up to the high side network manager so that the high side would then see both sides as two different machines. The low side of the BD can also be made to only respond to the high side network manager if it is not desired that the low side network manager "see" the BD.

The last two rules are added to deny all other possibilities. The 65000 rule, added by this proxy project, allows the logging of any denied packets to an audit trail. The 65535 is a default rule established by the rc.firewall script. The rule ensures the rejecting of any extraneous packages. Both are technically not needed, but this ensures what is happening and the logging function can easily be turned on and off.

```
ipfw add 65000 deny log ip from any to any
ipfw add 65535 deny ip from any to any
```

3.5.9 Does NOT directly route network traffic but instead proxy all packets

This requirement actually has two parts. The first is making sure that all packets go through the NETMON proxy daemons. The second is the actual proxying or rebuilding of a clean packet.

To meet the first requirement we just need to recap from a couple of the above sections. When in operation the boundary device subsequently diverts all UDP SNMP and certain types if ICMP packets that are allowed into the firewall to the proxy daemons that are listening on two distinct ports. At present and by default high2low and low2high listen on ports 9696 and 6969 respectively (see firewall rules 3.5.8). The 'divert' captures all the packets that are allowed to come into the firewall and sends those packets to the respective proxy daemon. The proxy daemon is responsible to ensure that the packets are either type UDP SNMP or are of type ICMP, content checking, and verifying against the SNMP security policy.

The second requirement is not entirely fulfilled. As a result of the screening process and the extent of the network address translation (snmp data field sanitization) most of the high to low SNMP packets are rebuilt. There is still a deficiency in the process for one to call what we are doing a full true proxy. Those packets that do not get sanitized, low to high for instance, are only repackaged for the network address translation. The original packets are not directly routed to their final destination but are repackaged because of the network address translation performed by the BSD Packet Filter (BSD).

BPF is a kernel facility that was originally developed for capturing and filtering network traffic. This facility was convenient for the prototype boundary device because it gave direct network access. In this phase of development, the objective was to determine whether or not SNMP traffic could be "proxied" between high and low networks effectively. The advantage of BPF in this application was that it provided simple, direct access to the network outside of IP and IPFW. Therefore, packets could be built and/or modified and transmitted without having to worry about their interaction with IP and IPFW. Events that require the BPF are translating IP addresses to obscure the high side, adjusting checksums, and scrubbing data fields. On the other hand, BPF does not support capabilities such as fragmentation or routing. Although, to date, these capabilities have not been required for proper operation of the boundary device.

3.5.10 Design to be automatic

The Netmon device is designed to boot up into operational mode with real-time packet filtering. It does NOT require a human to be logged onto the system to operate, screen or release the packets. The system is designed that if the health status of the interface control device becomes unstable it will shut the firewall and proxies down. The only human intervention on the device itself is to turn the system on, shut the system down, configure the policies, off-load or analyze the audit trail, and maintenance.

HP Openview does require human intervention in that one must be logged onto the system in order display the common operational picture.

3.5.11 Boundary Device must reside in a protected environment

One of security requirements of an interface control device is that the device be in the physical control of the user with the highest level of protection requirement. This is referred to as operating at “System High”. Operating at this level alleviates several very cumbersome security requirements.

3.5.12 Handle and Minimize covert channels

The basic premise to handle covert channels such as timing and sequencing is to reduce or eliminate the ability for someone to be able to sniff the network where the information is being broadcast. As discussed before there are some classic ways of programming solutions, adding random times, work on multiple requests and so on. These classic ways do work but work against our goal of keeping the device filtering real time. The project has addressed the problem through the use of an architecture that can layer security and eliminate the potential threat of someone eavesdropping. The following two figures will demonstrate the thought process that was used to come to our final architectural requirement.

The first step was handling timing issues. This can be resolved by inserting the NETMON device into the typical hierarchical network management architecture (Figure 3) and coding delays, heavy monitoring, keeping track of the number of requests and so on. Unfortunately, this architecture does not prevent someone on the low side or high side from being able to snoop and view network traffic for sequencing channels.

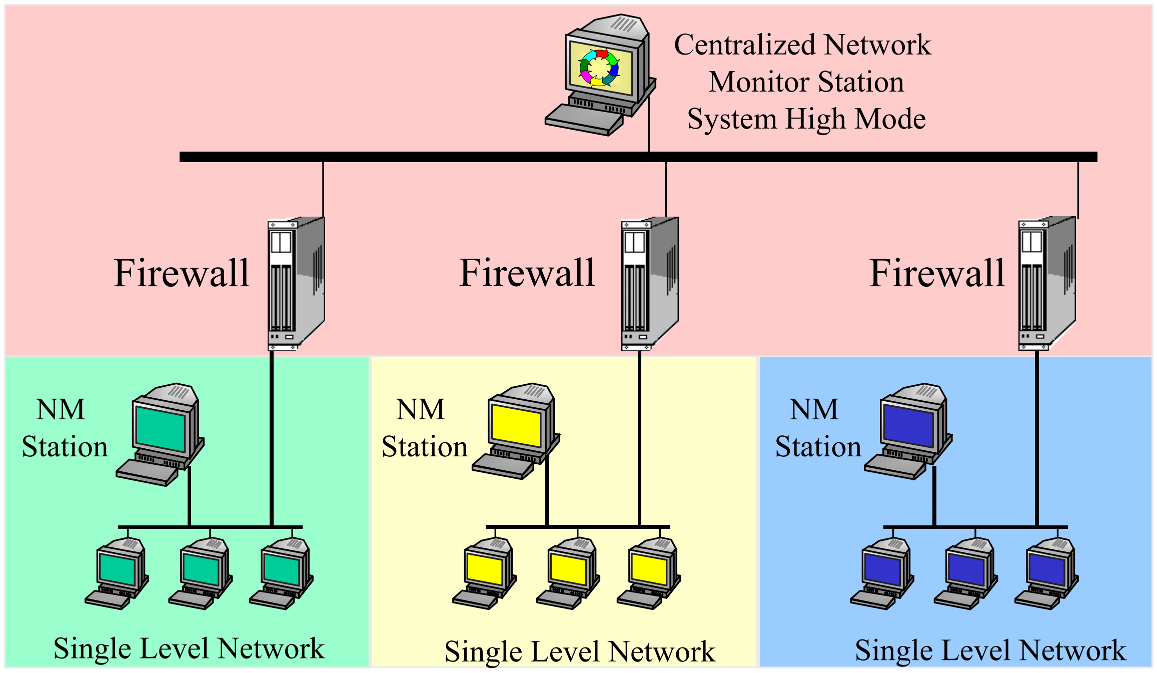

Figure 4: Architecture that could solve timing channels

To solve this problem on the high side a switch can be added to prevent viewing of information as shown in Figure 5. Another solution is to have the Central NMS have a Ethernet connection per interface control device (NETMON) being used. Configuring the Central NMS to not allow forwarding of packets to other network and create different networks segments to each device.

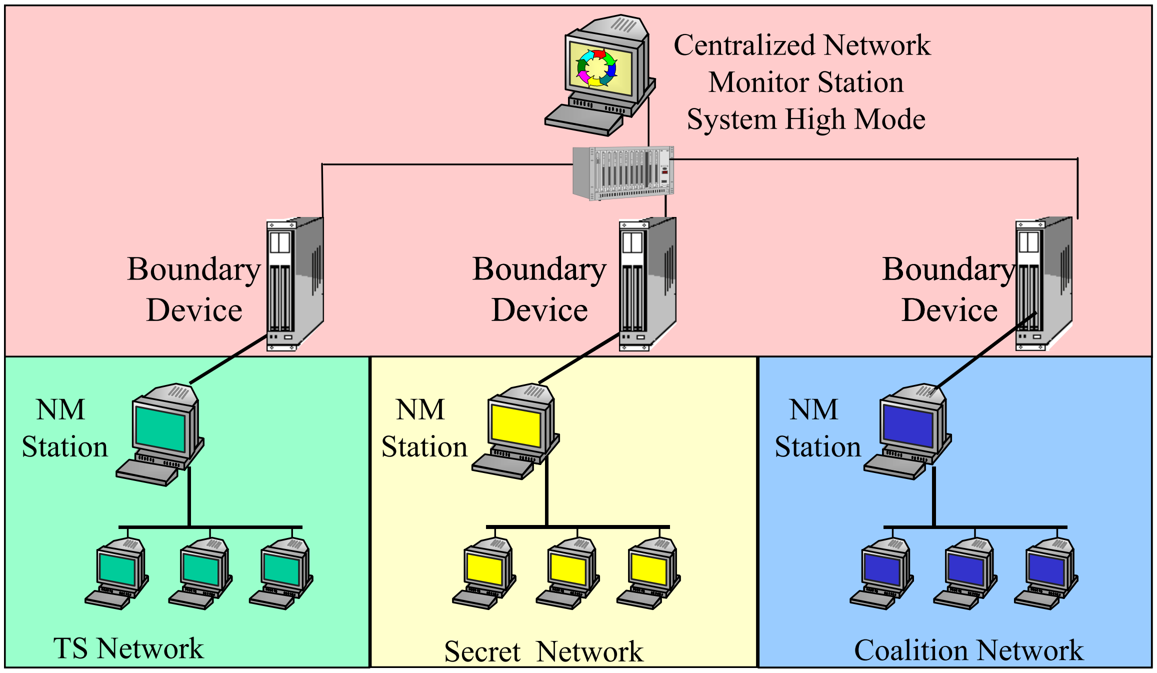

Figure 5: NETMON Final Architecture

The alternative solution to the high side is the exact solution for the low side. Each domain's NMS needs to have a separate Ethernet connection for the domain it is collecting information on and another connection for the reporting of domain status to the Central NMS. See Figure 5.

The final architecture selected for this effort is shown in Figure 5. The system uses a switch in the high side and a separate connection to the low side NMS. The advantages to this point far out weighed the disadvantages.

Advantages:

- Layered security approach continues to meet the COTS vs GOTS objective
- Has minimal impact on each participating domain requirement (1 Ethernet card, HP Openview)
- Completely isolates the heaviest network management traffic to a separate network minimizing operational network impact.
- Completely eliminates an enclave user from using his PC to snoop the network for sequencing and timing channels. If NMS is compromised, the NETMON device is configured to only communicate with the low side NMS. Another words it can only talk to one machine and will therefore not forward any other IP address which still prevents any type of sequencing channels.
- Allows for the addition of encryption devices for stronger point-to-point authentication. (NEC, VPN, KG)
- This architecture in conjunction with the current design does not allow an event on the high side to get a reaction from the low side domain. It is completely isolated to the low side NMS. The low side NMS does not forward information down to the low side or request information from the low side after receiving a request from the high side. The low side NMS only reports what it currently knows.
- Keeps the management responsibility in each domain. The low side NMS can do all the management and monitoring requirements at the same time as reporting to the central NMS.

Disadvantages:

- Currently limits the architecture for monitoring purposes only. Actual management of nodes in a different domain is not permitted. (This is an advantage for security purpose but a disadvantage to operational requirements.
- Enhancements to the overall system that would require the Central NMS to obtain information from a specific node will now require a process loaded onto the low side NMS to handle the requests. The current goal was to keep all coding modules strictly on the NETMON device.

3.5.13 Functional Description

This section outlines the handling process off the network management packets. Figure 6 is an effort to picture the handling process. The SNMP and ICMP sections that follow explain Figure 6. Each section refers to Step 1 through 4. These steps are shown with cubes with #1 through 4 in them. The cubes are over the process, which executes that particular function. For example Step 1 is a UDP packet hitting the firewall therefore the cube is on/near the brick wall.

Another point of interest to this figure is that all the code that has been developed resides in the circle between the firewall. Recall figure 2, the bullseye figure that showed the developed code in the center darker gray circle. The firewall and BPF are combined into the brick wall and is represented as the light gray (blue if viewing on the screen) application layer in figure 2.

The dotted line that splits the circle in half, red and black, is there to help visualize the security aspects. The firewall is not a MLS system and therefore does not truly have a separation between security levels like Trusted Solaris. This code may need to be ported to a MLS operating system and this figure would then be an accurate representation of where the processes would reside.

3.5.13.1 Current SNMP Handling:

Step 1: Firewall rules only allow UDP & ICMP to pass
Step 2: UDP filter (SNMP uses UDP as it's transport protocol)

- Source and destination check
- Protocol format checking
- Flow control/Type checking

Step 3: IP Translation to hide the actual High Network

- Low to High -> the Destination IP and MAC is changed to reflect actual High NMS's Ethernet IP and MAC
- Low side of Firewall appears as the High side NMS
- High to Low -> the Source IP and MAC are changed to reflect actual firewall's low side Ethernet IP and MAC
- High side of firewall appears as a gateway/router to High NMS

Step 4: BPF writes new packets out onto network

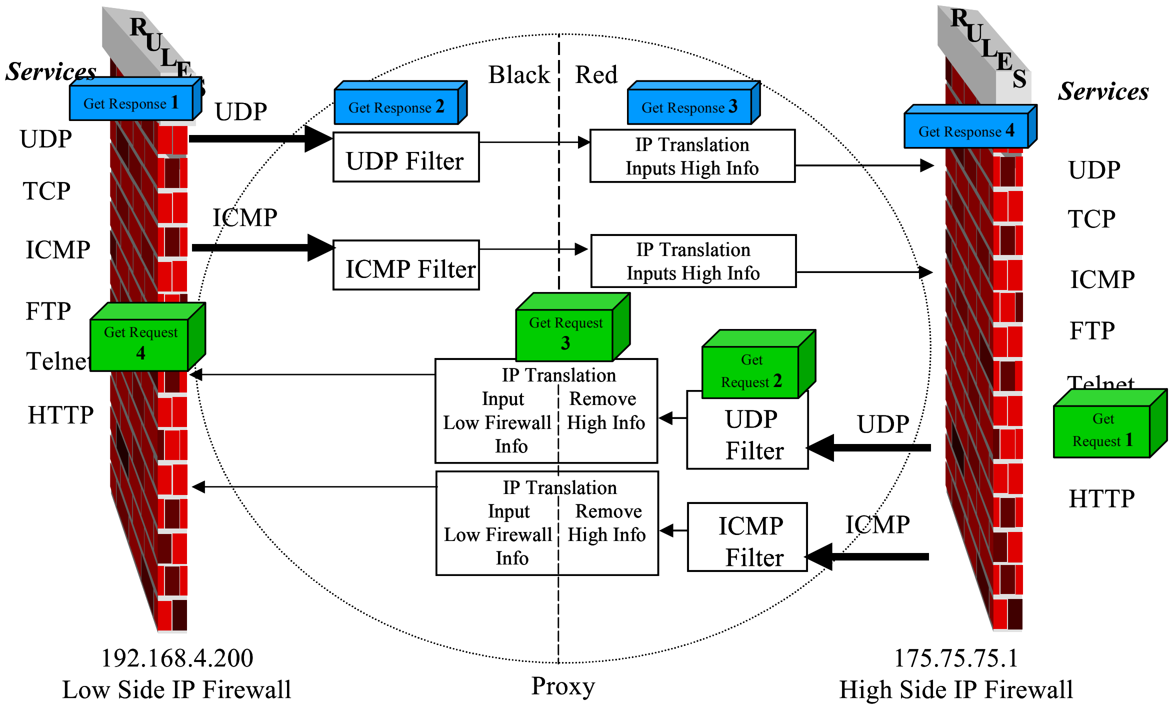

Figure 6: NETMON Device Functional Description

3.5.13.2 Current ICMP handling:

Step 1: Firewall rules defined to pass only UDP & ICMP

- Flow control/Type Checking (Echo Request/Response and Netmask Request/Response)

Step 2: ICMP Filter

- Source and destination check (Proxy)

Step 3: IP translation to hide the actual High Network

- Low to High -> the Destination IP and MAC is changed reflect actual High NMS's Ethernet IP and MAC
- Low side of Firewall appears as the High side NMS

• High to Low -> the Source IP and MAC are changed to reflect actual firewall's low side Ethernet IP and MAC

• High side of firewall appears as a gateway/router to High NMS

Step 4: BPF writes new packets out onto network

3.5.14 Coding

The programming language for the proxy is C. In the later stages of the development, an effort was put forward to create a user interface which was written in Java.

The development of NETMON from this point forward was done using the spiral development model, i.e build a little, test, test output used as input back into build. Milestones were only considered complete after thorough testing in the test facility at RL.

The major milestones that the development team needed to meet were:

- getting the proxy to recognize SNMP,
- getting the proxy to handle the format checking,
- getting the proxy to accomplish content checking,
- integrating the device with HP Openview,
- getting the proxy to search and replace high-side information, hostname and IP address) that HP Openview transmits when using the hierarchical network management capabilities.

4. Current Status

4.1 Testing

In depth testing was done throughout the course of the development. The testing took on several forms and was accomplished by several groups.

Arca Systems continually was testing in the test facility at RL. As described in an earlier section, before a milestone was considered complete a thorough testing had to be successfully passed. We developed our own test script to allow for reproducibility and consistency in testing. Appendix C has examples of the captured data from the high and low side network and the boundary device itself. The information has been presented in a logical order and color-coded as to where the information was captured. An independent team from ARCA and Exodus also conducted penetration testing on NETMON at the end of the development cycle. Appendix D has the output of that test.

BAE Systems was used by RL to provide an independent analysis of the device and system management support for the development and test facilities. BAE used a test program that they developed to provide RL and ARCA with a second verification of functionality. BAE Systems also provided an independent penetration test accomplished during the JEFX 00 demonstration (see next section). A final technical report was submitted by BAE for the JEFX effort.

4.2 Demonstrations

There were several demonstrations done throughout the course of the development. The demonstrations ranged from coalition representatives, operational groups, commanders, and senator aides. The most influential to the development of this effort were the collaboration effort demonstrated in Australia and the Joint Expeditionary Forces Exercise (JEFX) 2000. The following two figures show and briefly summarize the above two demonstration.

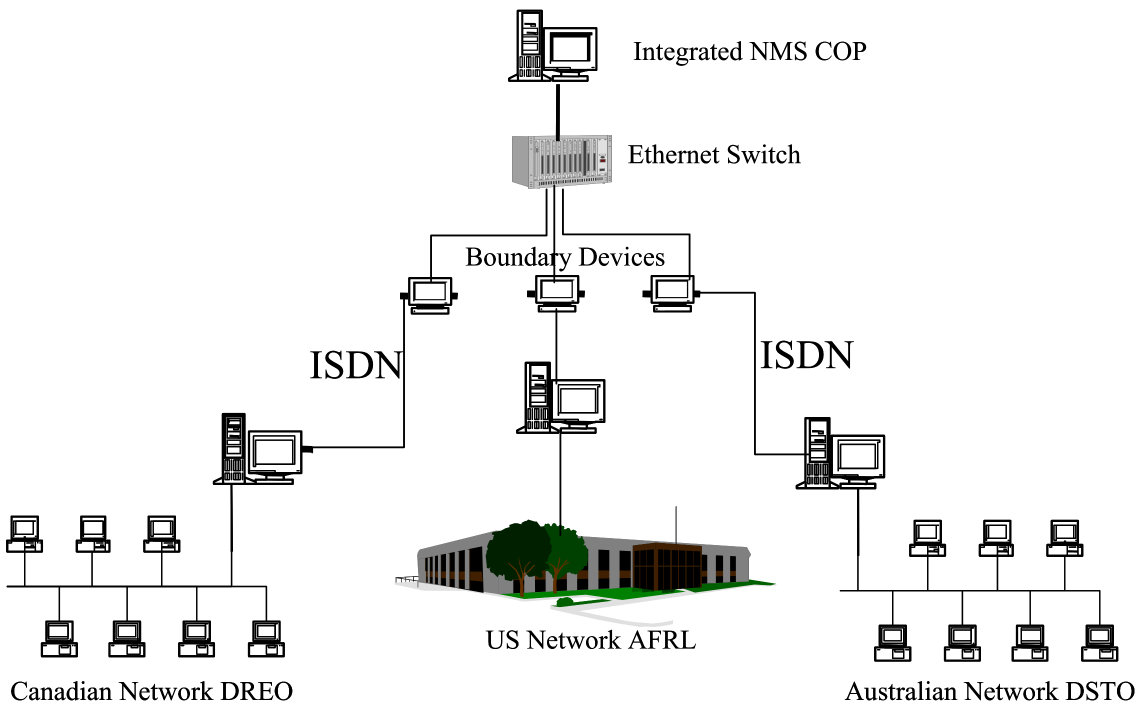

Figure 7: Architecture used for the Australian Demonstration.

NETMON was implemented as a Coalition Network Management Tool between three separate network testbeds in Ottawa Canada, Rome NY and Salisbury Australia in November 1999. The demonstration occurred as part of the international "The Technical Cooperation Panel (TTCP)" panel meetings in Salisbury Australia. Representative from AFRL RRS presented and demonstrated a multi national network management capability to representatives from Australia, New Zealand, Canada, United Kingdom and the US. The NETMON system operated in real time over the ACCORD ISDN infrastructure. Asset information from the three networks was combined into a network common operational picture to show a coordinated display of all three countries network testbeds. The Network COP was remotely displayed from Rome NY to the conference facilities at DSTO in Salisbury Australia. Machines in Rome and Salisbury were randomly disconnected and reconnected to the network. Within approximately 30 seconds, the Network Common Operational Picture correctly reflected the change in status of each asset. Integrity of the individual nations domains was maintained and controlled by individual specialized network management boundary devices connecting each nations network. The boundary devices resided in a controlled facility in Rome NY. The devices were developed as part of the MDNM program. The international audience of more than 60 scientists and engineers responded very positively to the coordinated demonstration and as a result, two separate cross fertilization meetings occurred between TP-8 (Technical panel for networking and communications) and other TP's to discuss collaborative work and how MDNM and the ACCORD network could be utilized and built upon for other coalition programs.

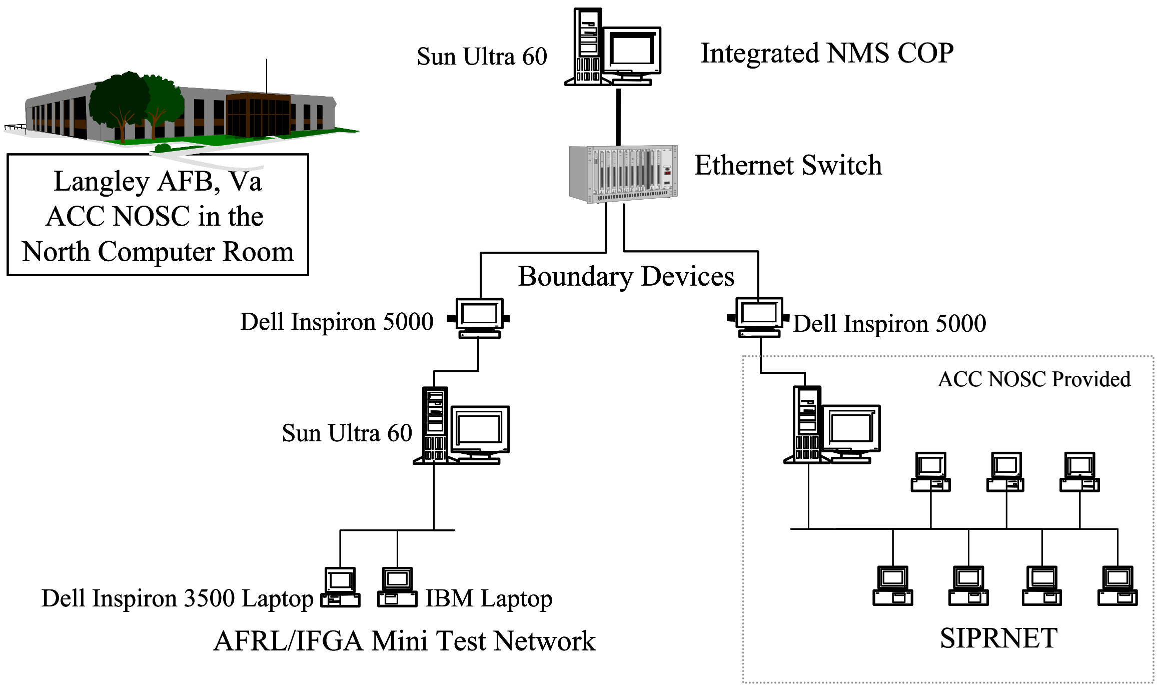

Figure 8: Architecture for the Joint Expeditionary Forces Exercise (JEFX) 2000

The development team and RL personnel integrated the NETMON devices, Centralized NMS, and a test network into the Air Combat Commands (ACC) Network Operations Systems Center, at Langley Air Force Base, Virginia. NETMON was implemented as a Network Management Tool between two separate networks; the JEFX secret network (SIPRNET) and a test network (emulating NIPRNET). The demonstration occurred as part of the Joint Expeditionary Forces Exercise in 2000. Asset information from the two networks was combined into a network common operational picture to show how a coordinated display could improve the system administrator's job in the NOSC. This demonstration, which lasted 2 weeks, gave the development team a first hand look at what tools the system administrators use, the environment in which they work, and a larger appreciation for the significance of that product like NETMON could bring to helping the administrators perform their functions. It also gave the administrators a chance to see what a product like NETMON could significantly help their efforts.

4.3 Enhancement Recommendations

This section will describe enhancements that our team feels are necessary for the continued success and transition of this product to the military as well as to the commercial enterprise.

4.3.1 SNMP Parser Redesign Requirement

The SNMP parser needs to be redesigned for a number of reasons:

1. Of all the boundary device implementation code the parser is and will continue to be the most portable piece of code. This is due primarily to the fact that it is at a high enough level in the protocol stack that it does not have to be concerned with issues involving transmission and reception of datagrams.
2. Presently, the parser works correctly, but since it was built for use in a prototype system it was not designed for portability, extensibility, or completeness.
3. It has been observed and confirmed that certain PDUs traveling from high to low reveal high side hostnames and IP addresses to low side NMSes. The fix to this problem was to build an additional routine/check that scrubs the variable binding field and replace appropriate information when necessary. Again the code works correctly, but it does not lend itself to expandability to encompass the next situation that may arise due to a change of NMS software.

4. The addition of new protocols, ICMP and SNMPv3, could be more easily handled with a better implementation.
5. Portability of code to a trusted operating system. Currently, the code is written modularly but not written in such a way that it is easily modified to work in an MLS or Secure Operating system.
6. It can be written in a way that would support a true proxy function.

Because the final parser stands a very good chance of being highly portable, it makes sense to put the effort into a redesign. If done properly, the new parser could realistically have the following attributes:

- Portability
- Extensibility
- Efficiency
- Completeness
- Correctness

4.3.2 Error Handler Redesign Requirement

Error handling is currently in need of redesign. The current thought is that any time an error occurs in a major subsystem of the proxy daemon, it should be delivered to an error handler that can take the appropriate action based on a predetermined configuration. This centralized management of error conditions has several advantages:

- errors can easily be counted and tracked
- formatting of error messages can be easily changed
- the amount of debugging information can be easily changed
- its easier to change the disposition of the daemon inside a consolidated error handler than from where the error actually occurs

4.3.3 True Proxy Requirements

As discussed in the design and build section, the NETMON device needs to truly proxy all packets whether or not a data field has been modified. This requirement would also include the removal of the dependency of the Berkley Packet Filter for portability requirements.

4.3.4 Maintaining State

It should be possible to construct tables based on SNMP and ICMP requests. The goal from this idea is to prevent the ability to send a response through the NETMON device without having had a Request go through.

4.3.5 Addition of ICMP into the format and content checking

To ensure this product can securely handle the network management traffic all protocols should meet the same requirement that was developed for the SNMP. Arca believes that would include ICMP even though the firewall has proved reliable. By incorporating this protocol into the proxy daemons it further reduces the risk of anything getting through even with a misconfiguration of the firewall.

4.3.6 Addition of SNMPv3

This is needed for completeness of the SNMP suite. This introduces the complexity of encryption and how to screen packets with the information encrypted.

4.3.7 Further Development of the User Interface

The current GUI has proven extremely useful but needs further work to make complete. The goal for the user interface was to provide a single point of configuration for the system administrator, Health verification tools, Audit reduction tools, and other maintenance capabilities.

4.3.8 Integrate with another major enterprise network management software package.

One of the objectives of this effort has been independence of the network management program. Effort has been made to ensure this with using available packages that do not support hierarchical management capabilities. A formal effort should be made to use an enterprise management program that does support hierarchical management such as Tivoli.

4.3.9 Management of nodes vs Monitoring

The current product can only be used to provide a monitoring capability. Users have requested the ability to control devices and get link status from a central location as well. This requirement is a much larger than it appears. This affects all aspects that have been created thus far. The architecture, the writing of network management modules that would need to reside on the NMS, can possibly open covert channels, a special SetRequest filter, and a stronger variable bindings field screener/filter are just a few of the areas touched by this needed requirement.

4.3.10 Hardware Enhancements

The following list is a goal that Arca's development team feel could be accomplished over time.

Testing of other Firewalls (Sidewinder and Gauntlet)
Reduction in size from laptop to cable modem or boca hub size.
Reduction to a card.

4.3.11 Operating System Enhancements

The following list of operating systems is NOT a promotion to port the NETMON application to them. It is however a list of possible candidates that should be continually evaluated to ensure success in different security realms that NETMON may need to support.

Trusted BSD
Trusted LINUX
Trusted Solaris

4.3.12 Integration of this work with DIW tools

To provide a cohesive enterprise management system NETMON or some portion of the technology could be integrated into DIW tools. The integration of this effort could take on the form of passing system logs to an analyzer or traps being sent to an agent when a break in attempt has been made. It is unknown at this point the exact nature of the integration but it is known that information that comes from a boundary device can is essential in assessing the aftermath of an attack.

4.4 Conclusion

The current effort comes to a close with a device that has proven to be successful in the real-time filtering of network management traffic. Attempts to spoof NETMON have failed and thorough testing substantiates the success of protocol handling. The product has also proven stable and reliable and has supported many demonstrations that have required the relocating and reconfiguring the equipment and short notice preparation.

ARCA believes that NETMON has met the initial goals set forth at the beginning of the effort. The NETMON device will boot into operational mode and operates without human intervention. The system administrator has a simple GUI tool that simplifies the configuration of the device and provides a simple audit viewing capability. All coding resides on the NETMON device itself. The architecture, though limits future enhancements, successfully mitigates covert channels such as timing and sequencing.

It is believed that the current architecture and design of the system could meet the Common Criteria EAL2 evaluation without further modification. The current design would not meet the protection level 4 that a boundary device attached to a top secret network would need as prescribed by the DCID 6/3. The current design would only meet a PL 2. Therefore, it was determined not to pursue the TOP SECRET arena at this time. Evaluation of the SABI evaluation process was ongoing and not complete. It is known that SABI does have a requirement that security products must be evaluated under the Common Criteria process. With any future work this effort will have to continually re-evaluate how the device and architecture stacks up against the various and ever changing security requirements that are being targeted (Common Criteria, SABI, DIA).

Appendix A: Firewall comparison Matrix

Product	Cyberguard Firewall		NAI/TIS Guantlet 4.0		Secure Computing Sidewinder Security Server		Borderware Firewall Server		Microsoft Proxy Server		FreeBSD Internet Firewall	
Capabilities												
OS Platform	SCO UnixWare		Unix (BSD variant), Win NT, DMS, IRIX, Solaris, HP-UX		Secure OS, a modified BSD		Modified BSD		Win NT		FreeBSD	
Multi-Level Support	X					See advantages						
Evaluated Platform (History of…)	X	Previous versions of firewall receive B1	X	ITSEC E3								
Proxy Server	X	No SNMP Proxies	X	Based on Firewall Toolkit Application Proxy	X		X	No SNMP Proxies	X	Packet Filtering and Packet Proxying		IP Firewall is application level gateway
Generic Proxy	X	IP Port control	X		X		X	Auto-Proxy: auto-configures new services				Proxies can be implemented but do not exist by default
Packet Field Filtering												
Platform Development Tools												
- OS	X	SCO Dev. Kit	X	BSD Dev tools	X	BSD Dev tools	X	BSD Dev tools	X	Windows SDK	X	BSD Dev tools
- Firewall		Requires vendor relationship	X	Development based on open source code		Requires vendor relationship		Requires vendor relationship			X	Open Source Code
Y2K Compliance	X		X		X						X	
DII COE												
SNMP Manageable	X		X		X				X	If agents are installed		Only if agents are installed
Advanced Management Interface	X	Local and secure remote management	X		X	Remote & Centralized Management	X	HTTP Based management interface	X	Web Based Administra-tion		
Interface Adaptability			X	Java Based GUI	X	Java based interface						N/A
Active Content Filtering	X	Virus Safe	X		X				X			
Development tools for Active Filtering												
Packet Flow Control	X		X		X		X		X		X	
Network Address Translation	X		X		X	Source and Destination Port re-mapping	X		X		X	
Split DNS	X		X		X		X		X		X	
VPN	X	Stoplock Connect, 3rd Party supplier	X	Global VPN, Client level VPN	X	Integrated, Client Level VPN	X	Dynamic VPN				
Protocol Support	\	No supported capability.					\	No supported capability.		Only through		Protocols are

		Protocol can be limited through packet filtering.						Protocol can be limited through packet filtering.		SOCKS complianc e		supported by installation of protocol elements
- SNMP			X	Proxy	X	Application -level gateway					X	
- SNMP v2			X	Proxy	X	Application -level gateway					X	
- RMON											X	
- RMON v2											X	
Advantageous Capabilities	- Initial Relationship with vendor - Availability of Generic Proxy Source Code through relationship - Multi-level OS		- Stalker 3.0 – intrusion detection capability - BSD Based development environment		- Strikeback Response - Type Enforcement provides process separation that emulates MLS functionality - SNMP Attack Detection - BSD Based Development environment						- Inexpensive (free) - Open development environment - Freely available SNMP and Proxying code - Installable support for multiple protocols - Easy portability	
Disadvantages	- Poor vendor support in development - OS Specific develop requirements - Lack of portability in code.		-									

Appendix B: Network Monitoring Software Matrix

The following table is a comprehensive list of commercial vendors and their products that deal with Network Monitoring. Though ARCA researchers have updated and corrected items in this list, the main list itself was taken from the Technical University of Twente's web page **http://snmp.cs.utwente.nl/software/commercial.html**. This project will try and take advantage of some of the free demo and Beta versions that these commercial assets provide in order to test cross-compatibility of software with our development effort. An effort was made to categorize these products by type of monitoring (application, network only, enterprise, developer kit), demo available, and provide OS support for those of type Network Monitoring. Information on some of the web pages was insufficient to answer some of the questions.

Company	Cntry	Software	App or Network Monitor	OS Support	Demo Avail	Web
AppliScope	USA	A software tool which monitors application programs.	Application			www.appliscope.com
ARINC	USA	Taboret SNMP Management Application Builder.	Developer Kit			www.taboret.com
Bay Networks, Inc	USA	Optivity Network Management - Various network management products.	Network	NT 4.0/95 Unix	N	www.baynetworks.com
Banyan Systems, Inc	USA	NetWizard - Distributed Software Management.	Application			www.banyan.com
Bull	USA	OpenMaster by BullSoft	Enterprise (inc Network)	Not listed		www.openmaster.com
Cabletron Systems, Inc	USA	Spectrum.	Network	NT 4.0 PAK3 Solaris 2.5.1 & 2.6	N	www.ctron.com/spectrum
Caravelle	CAN	Trinity Suite	Enterprise (inc Network)	NT 4.0 Solaris2.5.1 Client: includes 95 & 98	N	www.caravelle.com
Castle Rock Computing	USA	SNMPc	Network	NT	Y	www.castlerock.com
Cisco	USA	Enterprise network management	Cisco Network Only			www.cisco.com
ClearSystems, Inc.	USA	ClearStats	Network	NT 4.0, HP-UX 10.10, 10.20 and SUN Solaris 2.5x.	N	www.clearstats.com
COMPU-SHACK	DE	Active CS-Care	Novell Network Only			www.cscare.com/cscare
Olicom / Crosscomm	USA	Clearsight Network Management System	Network	NT & 95 Optional HP Openview for windows	N	www.crosscomm.com
Digital Equipment	USA	TeMIP	Enterprise (inc Network)	Possible NT support UNIX (info unclear no specs online)	N	www.digital.com
Empire Technologies, Inc	USA	Systems Management agents and management station applications.	Network	NT 3.51/4.0 on x86 NT 4.0 on Alpha Sun 4.1.x Solaris 2.x on Sparc Solaris 2.x on x86 HpUx 9 & 10.x IRIX 6.x AIX 4.x Digital Unix 4.x Linux 2.x(1Q99)	N	www.empiretech.com
Hewlett Packard Company	USA	OpenView. (Network Node Manager is basic Network Monitoring module)	Enterprise (inc Network)	NT 4.0 HP-UX 10.20, 11 Solaris 2.5.1, 2.6	N	www.hp.com
International Business Machines,	USA	Nways - Management solutions for Networking. Appears to be	Networking (various		N	www.networking.ibm.com

Corporation		IBM specific	components)			
ISR Global Telecom	USA	Object-oriented TMN products.	Developer Kit			www.isrglobal.com
Kaspia Systems, Inc.	USA	Automated Network Monitoring System (ANMS).	Networking	NT 4.0 PAK 3	N	www.kaspia.com
Kinnetics (Loran Technologies)	CAN	Kinnetics Enterprise Network Manager (SNMP / JAVA)	Networking	Propriety Hardware/Software Integration	N	www.kinnetics.com
LMC (LAN Management Consulting	DE	CINeMa (Computer Integrated Network Management)	Networking	NT, 95	Partial	www.Imc.de
Micromuse	UK	Netcool/Omnibus (service provider management)	Networking	UNIX	N	www.micromuse.co.uk
MG-SOFT Corporation	SI	MG-WinSNMP SDK, MONET SNMP analyzer. (NetInspector)	Networking Developer Tools	NT & 95	Y	www.mg-soft.si
N.E.T	USA	PanaVue management Platform.	Networking	solaris 2.5.1 need HP OpenView	N	internet.net.com
NetIQ	USA	NetIQ AppManager Suite.	Windows NT environment only			www.netiq.com
Novell	USA	ManageWise.	Network	95, 98, & NT	N	www.novell.com
Nulink	India	ViewLAN (NT software, Web-based).	Network	NT	Y 30d	www.nulnk.com
Objectstream	USA	Mediator.	Developer Kits			www.objectstream.com
Optical Data Systems	USA	InfiniteView	Network	UNIX oracle 7 workgroup HP Openview	N	www.ods.com.
OutBack Resource Group, Inc.	USA	jSNMP Enterprise, a Java/CORBA development tool for distributed SNMP applications..	Developers Kits Admin Tools			www.outbackinc.com
Proteon LAN Products by Microvitec Inc	USA	LANtracer (Web based LAN monitoring)	Network	NT 4.0	Y	www.lantracer.com
Seagate	USA	NerveCenter	Network	NT Solaris v2.5.1 or HP/UX v.10.20	N	www.sems.com
Siemens/Nixdorf AG	DE	TransView	Enterprise (inc. Network)	NT & 95 UNIX		www.siemensnixdorf.com/servers/tv/tv_us.htm
Simware, Inc	USA	REXXWARE	Developer Kit for Novell			www.simware.com
SNMP Research, Incorperated	USA	EMANATE (Mainly developers Kit)	Network and Developers Kit	NT Most UNIX	N	www.int.snmp.com
Solcom Systems	UK	RMON Network Management Systems.	Network	NT Sun - Solaris 1 & 2 IBM - AIX DEC - OSF HP - UX	N	www.solcom.co.uk
Sun Microsystems (SunSoft)	USA	Solstice Enterprise Management	Network and Enterprise	SolarisTM 2.4 environment or later for x86 Solaris 2.4 environment or later for SPARC Solaris 1.1.1 (SunOSTM 4.1.3) environment or later	N	www.sun.com
Trent Datacomms	UK	MeterWare	Network	95 or NT 4.0	Y	www.trent.co.uk
3Com, Inc	USA	Transcend and Web Based management	Enterprise and Network	95 & NT UNIX	N	www.3com.com
Tivoli Systems, Inc	USA	Tivoli Management Environment (TME)	Enterprise	DG-UX, HP-UX, IBM AIX, Motorola SVR4, Solaris, SunOS, Windows	N	www.tivoli.com

				NT		
Wandel & Goltermann	DE	QMonitor	Network tools No mention of SNMP		N	www.qmonitor.wg.com
Wipro	India	CyberManage (Web-BasedManagement)	Network	95 & NT 4.0 Solaris 2.6 with JDK 1.1 support.	Y	cybermanage.wipro.com

Table 2: Commercially Available Network Management Software

The Public Domain resources are mainly for developers and development environments. There are specialized tools that allow the user to walk MIBS, generate SNMP Traffic, capture and replay SNMP Traffic, and libraries. There are a few full Network Monitoring Packages.

Company or Organization	**Description**	**Web**
Advent Network Management JAVA SNMP Package	JAVA based SNMP protocol stack & Management tools	www.adventnet.com
Carnegie Mellon University (CMU)	Snmp software Library	www.net.cmu.edu
CyberPro International	snmx 3.5 - Simple Network Management Executive	www.cpro.com
Dartmouth University	SNMP Watcher: An SNMP MIB Browser	www.darthmouth.edu
D. Matthews	SMAP: Network traffic visualization tool	www.kagi.com
Equivalence MibMaster	SNMP Web based MIB Browser	www.ozemail.com.au
Gus Estrella	Network monitoring for Windows 95 and Windows NT 4.0	www.digitaldaze.com
HP WebReporter	Web based Network monitoring tool	www.tmo.hp.com
INS Enterprise PRO	Web based Performance management tool	www.ins.com
IntraSpection	free SNMP Web-based Network Management Software	www.intraspection.com
Lumos Hypermib	ITU TMN models in browseable form	www.lumos.com
Marvel	Marvel is a standalone web-based management system written entirely in Java.	www.research.att.com
Master Assignment "Web based Management"	Master assignment of a student of the Simple group	ftp.cs.utwente.nl
Merit	Netscarf: Network Statistics Collection And Reporting Facility	www.merit.edu
MG-SOFT Corporation	MG-WinSNMP SDK, Net*Inspector and MONET SNMP analyzer	www.mg-soft.si
Network Computing Technologies	SNMP Packet Library in C++, Trap Generator and Receiver, Thingy etc.	www.ncomtech.com
NetPrism	Java-based network management system and development environment for system administrators and device vendors.	www.
Monash University	SNMPY: Python interface to SNMP	www.rdt.monash.edu.au
ntop	A tool that shows the network usage, similar to what the popular top Unix command does.	www-serra.unipi.it
NVision	Java Based Management Tool	www.edge-technologies.com
Optimal Networks Internet Monitor	Web based monitoring tool	www.optimal.com
OSIMIS	OSI Management Information Service Platform	www.cs.ucl.ac.uk
Pohang University of Science and technology	Network Management Using Java	madonna.postech.ac.kr

SNMP Sniff	A network sniffer to solve SNMP problems.	elektra.porto.ucp.pt
SMB_SNMP	SMB_SNMP is an extension to the popular Samba system. It basically allows people to see SNMP agents as shared Windows resources.	jake.unipi.it
SUN JAVA Management API	specs and nice demos	java.sun.com
SWITCH	SYSMAN, SNMP LISP library	www.switch.ch
SWITCH	SNMPerl5, SNMP support perl 5.	www.switch.ch
Technical University of Delft	The Tricklet(RMON) snmpv1 package	dnpap.et.tudelft.nl
Technical University of Twente	Scotty, Damocles and other SNMP software.	wwwsnmp.cs.utwente.nl
Thinsoft	ThinSoft Ranger is an SNMP Manager program written entirely in Java.	www.thinsoft.com
Tobias Oetiker	Multi Router Traffic Grapher is a web-based tool to monitor the traffic load on network-links	ee=staff.ethz.ch
University of California	ucd-snmp v3.0, extensable SNMP agent	www.ece.ucdavis.edu
University of Florida	CMU-SNMPv2 toolkit for Linux	www.cis.ufl.edu
University of Fulda	Management tool with Java and Tcl	mnm.html at 193.174.26.169
University of Munich	The WILMA snmpv1/2 toolkit	ftp.ldv.e-technik.tu-muenchen.de
UTopia	Java web-based ATM Management package of the University of Twente Telematics Systems and Services management group	www.cs.utwente.nl
Web based Enterprise Management	Hypermedia management using HMMP	wbem.freerange.com
Webbin CMIP	Webbin project from Luca Deri (CMIP, SNMP and JAVA)	www.alphaworks.ibm.com
West Consulting BV	Rudimentary SNMP stack in JAVA	www.west.nl
Westhawk	SNMP stack in Java	www.westhawk.co.uk
The NAS Hierarchical Network Management System	X-Windows, SNMP based management software package	ftp.bayarea.net
WWW SNMP MIB Browser	CGI script that enables SNMP MIB browsing, based on SCOTTY	www.cs,tu-bs.de
ModularSNMP project	The goal of the project is to obtain a working Modular SNMPv3 engine in Java.	atm.teleinfo.uqam.ca
NocMonitor	A network Monitoring and Alert system written by Leland E. Vandervort. The system uses the PERL programming languague, and has HTML output.	www.discpro.org

Appendix C: Example Test Results

The following is the results of testing that was accomplished during the NETMON development. The output is captured data that resides either on the high-side NMS, the NETMON device, low-side device. The data was then put into a logical format and color-coded.

Example 1: All commands allowed through NETMON Device with detailed information looking into the high-side network and NETMON device:

KEY: Yellow High-Side NMS
Green is an allowed at the NETMON device
Red is a disallowed at the NETMON device
Blue is low-side NMS

```
 echo "TEST 1: HL: SETREQUEST V1"
 echo ""
/opt/OV/bin/snmpset -v 1 -d bass system.sysContact.0 octetstring "BOB
JONES"

Transmitted 50 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a3 23 02
00.....netmon.#.
    16:  02 40 2a 02 01 00 02 01 00 30 17 30 15 06 08 2b
.@*......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............

     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 16426
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[303002010004066e65746d6f6ea3230202402a02010002010030173015060682b06010201
0104000409424f42204a4f4e4553]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: SetRequest
Feb  6 13:45:46 myname high2low[317]:  SETREQ High to Low:
0202402a02010002010030173015060682b060102010104000409424f42204a4f4e45530
```

```
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 78 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 92
bytes.


Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 78 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0

Received 50 bytes from bass (192.168.3.2) port 161:
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a2 23 02
00.....netmon.#.
    16:  02 40 2a 02 01 00 02 01 00 30 17 30 15 06 08 2b
.@*......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e     .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............

     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    RESPONSE-PDU (0xa2): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 16426
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"


system.sysContact.0 : DISPLAY STRING- (ascii):  BOB JONES

 echo "TEST 2: HL: GETREQUEST V1"
 echo ""
/opt/OV/bin/snmpget -v 1 -d bass system.sysContact.0

TEST 2: HL: GETREQUEST V1

Transmitted 41 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 00 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 11 d7 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.........0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 4567
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes


Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[302702010004067075626c6963a01a020211d702010002010030 0e300c06082b06010201
01040005004]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: GetRequest
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[302702010004067075626c6963a01a020211d702010002010030 0e300c06082b06010201
01040005004]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 69 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 83
bytes.


Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 78 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0

Received 50 bytes from bass (192.168.3.2) port 161:
     0:  30 30 02 01 00 04 06 70 75 62 6c 69 63 a2 23 02
00.....public.#.
    16:  02 11 d7 02 01 00 02 01 00 30 17 30 15 06 08 2b
.........0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e     .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............
```

```
     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 4567
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"


system.sysContact.0 : DISPLAY STRING- (ascii):  BOB JONES


 echo "TEST 3: HL: GETNEXTREQUEST V1"
 echo ""
/opt/OV/bin/snmpnext -v 1 -d bass interfaces.ifTable.ifEntry.

TEST 3: HL: GETNEXTREQUEST V1

Transmitted 42 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 28 02 01 00 04 06 70 75 62 6c 69 63 a1 1b 02
0(.....public...
    16:  02 24 fd 02 01 00 02 01 00 30 0f 30 0d 06 09 2b
.$.......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9469
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes



Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[302802010004067075626c6963a11b020224fd020100020100300f300d06092b06010201
0202010005004]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: GetNextRequest
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[302802010004067075626c6963a11b020224fd020100020100300f300d06092b06010201
0202010005004]
```

```
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 70 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 84
bytes.

Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 72 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0

Received 44 bytes from bass (192.168.3.2) port 161:
     0:  30 2a 02 01 00 04 06 70 75 62 6c 69 63 a2 1d 02
0*.....public...
    16:  02 24 fd 02 01 00 02 01 00 30 11 30 0f 06 0a 2b
.$.......0.0...+
    32:  06 01 02 01 02 02 01 01 01 02 01 01 -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 42 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 29 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9469
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 17 bytes
    27:        SEQUENCE VARBIND (0x30): 15 bytes
    29:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.1
    41:          INTEGER (0x2) 1 bytes: 1

interfaces.ifTable.ifEntry.ifIndex.1 : INTEGER: 1

 echo "TEST 4: HL: TRAP (snmptrap) V1"
 echo ""
/opt/OV/bin/snmptrap -d bass "" "" 6 1 15 system.sysDescr.0
octetstringascii "SunOS hpcndnw2 4.1 1 sun4c"

TEST 4: HL: TRAP (snmptrap) V1

Transmitted 85 bytes to bass (192.168.3.2) port 162:
     0:  30 53 02 01 00 04 06 70 75 62 6c 69 63 a4 46 06
0S.....public.F.
    16:  0b 2b 06 01 04 01 0b 02 03 0a 01 02 40 04 c0 a8
.+..........@...
```

```
    32:  01 01 02 01 06 02 01 01 43 01 0f 30 28 30 26 06
........C..0(0&.
    48:  08 2b 06 01 02 01 01 01 00 04 1a 53 75 6e 4f 53
.+.........SunOS
    64:  20 68 70 63 6e 64 6e 77 32 20 34 2e 31 20 31 20      hpcndnw2
4.1 1
    80:  73 75 6e 34 63 -- -- -- -- -- -- -- -- -- -- --
sun4c...........

     0:  SNMP MESSAGE (0x30): 83 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 70 bytes
    15:      OBJ-ID ENTERPRISE (0x6) 11 bytes: .1.3.6.1.4.1.11.2.3.10.1.2
    28:      IPADDRESS AGENT-ADDR (0x40) 4 bytes: 192.168.1.1
(COPhost.ndf.af.mil)
    34:      INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    37:      INTEGER SPECIFIC-TRAP (0x2) 1 bytes: 1
    40:      TIMETICKS TIME-STAMP (0x43) 1 bytes: 15 (0xf)
    43:      SEQUENCE VARBIND-LIST (0x30): 40 bytes
    45:        SEQUENCE VARBIND (0x30): 38 bytes
    47:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    57:          OCTET-STR (0x4) 26 bytes: "SunOS hpcndnw2 4.1 1 sun4c"
```

```
Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[305302010004067075626c6963a446060b2b060104010b02030a01024004c0a801010201
0602010143010f302830260608 2b06010201010100041a53756e4f53206870636e646e773
220342e3120312073756e3463]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: V1_Trap
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[305302010004067075626c6963a446060b2b060104010b02030a01024004c0a801010201
0602010143010f302830260608 2b06010201010100041a53756e4f53206870636e646e773
220342e3120312073756e3463]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 113 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 127
bytes.
```

```
 echo "TEST 5: HL: TRAP (snmpnotify) V1"
 echo ""
/opt/OV/bin/snmpnotify  -d  -v 1 bass .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"

TEST 5: HL: TRAP (snmpnotify) V1
```

```
Transmitted 82 bytes to bass (192.168.3.2) port 162:
     0:  30 50 02 01 00 04 06 70 75 62 6c 69 63 a4 43 06
0P.....public.C.
    16:  09 2b 06 01 04 01 0b 02 11 01 40 04 c0 a8 01 01
.+........@.....
    32:  02 01 06 02 01 03 43 01 02 30 27 30 25 06 08 2b
......C..0'0%..+
    48:  06 01 02 01 01 01 00 04 19 53 75 6e 4f 53 20 65
.........SunOS e
    64:  63 6c 69 70 73 65 20 35 2e 33 20 31 20 73 75 6e     clipse 5.3 1
sun
    80:  34 6d -- -- -- -- -- -- -- -- -- -- -- -- -- --
4m..............

     0:  SNMP MESSAGE (0x30): 80 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 67 bytes
    15:      OBJ-ID ENTERPRISE (0x6) 9 bytes: .1.3.6.1.4.1.11.2.17.1
    26:      IPADDRESS AGENT-ADDR (0x40) 4 bytes: 192.168.1.1
(COPhost.ndf.af.mil)
    32:      INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    35:      INTEGER SPECIFIC-TRAP (0x2) 1 bytes: 3
    38:      TIMETICKS TIME-STAMP (0x43) 1 bytes: 2 (0x2)
    41:      SEQUENCE VARBIND-LIST (0x30): 39 bytes
    43:        SEQUENCE VARBIND (0x30): 37 bytes
    45:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    55:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"

Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[305002010004067075626c6963a44306092b060104010b0211014004c0a8010102010602
010343010230273025060082b06010201010100041953756e4f532065636c6970736520352
e3320312073756e346d6]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: V1_Trap
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[305002010004067075626c6963a44306092b060104010b0211014004c0a8010102010602
010343010230273025060082b06010201010100041953756e4f532065636c6970736520352
e3320312073756e346d6]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 110 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 124
bytes.

echo "TEST 6: HL: SETREQUEST V2"
 echo ""
/opt/OV/bin/snmpset -v 2c -d bass system.sysContact.0 octetstring "Herb
Markle "
```

```
TEST 6: HL: SETREQUEST V2

Transmitted 53 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a3 26 02
03.....netmon.&.
    16:  02 68 84 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.h.......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 26756
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "


Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[303302010104066e65746d6f6ea32602026884020100020100301a301806082b06010201
010400040c48657262204d61726b6c65200]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: SetRequest
Feb  6 13:45:46 myname high2low[317]:  SETREQ High to Low:
02026884020100020100301a301806082b0601020101040004 0c48657262204d61726b6c6
5200
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[303302010104066e65746d6f6ea32602026884020100020100301a301806082b06010201
010400040c48657262204d61726b6c65200]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 81 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 95
bytes.


Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 81 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0


Received 53 bytes from bass (192.168.3.2) port 161:
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a2 26 02
03.....netmon.&.
    16:  02 68 84 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.h.......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    RESPONSE-PDU (0xa2): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 26756
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "


system.sysContact.0 : DISPLAY STRING- (ascii):  Herb Markle

 echo "TEST 7: HL: GETREQUEST V2"
 echo ""
/opt/OV/bin/snmpget -v 2c -d bass system.sysContact.0

TEST 7: HL: GETREQUEST V2

Transmitted 41 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 45 e3 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.E.......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 17891
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes
```

```
Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[302702010104067075626c6963a01a020245e3020100020100300e300c06082b06010201
01040005004]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: GetRequest
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[302702010104067075626c6963a01a020245e3020100020100300e300c06082b06010201
01040005004]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 69 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 83
bytes.


Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 81 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0

Received 53 bytes from bass (192.168.3.2) port 161:
     0:  30 33 02 01 01 04 06 70 75 62 6c 69 63 a2 26 02
03.....public.&.
    16:  02 45 e3 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.E.......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 17891
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "


system.sysContact.0 : DISPLAY STRING- (ascii):  Herb Markle
```

```
 echo "TEST 8: HL: GETNEXTREQUEST V2"
 echo ""
/opt/OV/bin/snmpnext -v 2c -d  bass system.sysContact.1

TEST 8: HL: GETNEXTREQUEST V2

Transmitted 41 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a1 1a 02
0'.....public...
    16:  02 13 c0 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.........0.0...+
    32:  06 01 02 01 01 04 01 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 5056
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.1
    39:          NULL (0x5) 0 bytes


Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[302702010104067075626c6963a11a020213c002010002010030 0e300c06082b06010201
01040105004]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: GetNextRequest
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[302702010104067075626c6963a11a020213c0020100020100300e300c06082b06010201
01040105004]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 69 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 83
bytes.


Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 73 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0

Received 45 bytes from bass (192.168.3.2) port 161:
     0:  30 2b 02 01 01 04 06 70 75 62 6c 69 63 a2 1e 02
0+.....public...
    16:  02 13 c0 02 01 00 02 01 00 30 12 30 10 06 08 2b
.........0.0...+
    32:  06 01 02 01 01 05 00 04 04 62 61 73 73 -- -- --
.........bass...

     0:  SNMP MESSAGE (0x30): 43 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 30 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 5056
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 18 bytes
    27:        SEQUENCE VARBIND (0x30): 16 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.5.0
    39:          OCTET-STR (0x4) 4 bytes: "bass"


system.sysName.0 : DISPLAY STRING- (ascii):  bass

 echo "TEST 9: HL: GETBULKREQUEST V2"
 echo ""
/opt/OV/bin/snmpbulk -v2c -d  bass interfaces.ifTable.ifEntry.0

TEST 9: HL: GETBULKREQUEST V2

Transmitted 42 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 28 02 01 01 04 06 70 75 62 6c 69 63 a5 1b 02
0(.....public...
    16:  02 26 f8 02 01 00 02 01 05 30 0f 30 0d 06 09 2b
.&.......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETBULK-REQUEST-PDU (0xa5): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9976
    19:      INTEGER NON-REPEATERS (0x2) 1 bytes: 0
    22:      INTEGER MAX-REPETITIONS (0x2) 1 bytes: 5
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
```

```
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[302802010104067075626c6963a51b020226f8020100020105300f300d06092b06010201
0202010005006]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: GetBulk
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[302802010104067075626c6963a51b020226f8020100020105300f300d06092b06010201
0202010005006]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 70 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 84
bytes.


Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 145 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0

Received 117 bytes from bass (192.168.3.2) port 161:
     0:  30 73 02 01 01 04 06 70 75 62 6c 69 63 a2 66 02
0s.....public.f.
    16:  02 26 f8 02 01 00 02 01 00 30 5a 30 0f 06 0a 2b
.&.......0Z0...+
    32:  06 01 02 01 02 02 01 01 01 02 01 01 30 0f 06 0a
............0...
    48:  2b 06 01 02 01 02 02 01 01 02 02 01 02 30 0f 06
+............0..
    64:  0a 2b 06 01 02 01 02 02 01 01 03 02 01 03 30 11
.+............0.
    80:  06 0a 2b 06 01 02 01 02 02 01 02 01 04 03 6c 6f
..+...........lo
    96:  30 30 12 06 0a 2b 06 01 02 01 02 02 01 02 02 04
00...+..........
   112:  04 68 6d 65 30 -- -- -- -- -- -- -- -- -- -- --
.hme0...........

     0:  SNMP MESSAGE (0x30): 115 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 102 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9976
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 90 bytes
    27:          SEQUENCE VARBIND (0x30): 15 bytes
    29:            OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.1
    41:            INTEGER (0x2) 1 bytes: 1
    44:          SEQUENCE VARBIND (0x30): 15 bytes
    46:            OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.2
    58:            INTEGER (0x2) 1 bytes: 2
    61:          SEQUENCE VARBIND (0x30): 15 bytes
    63:            OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.3
    75:            INTEGER (0x2) 1 bytes: 3
    78:          SEQUENCE VARBIND (0x30): 17 bytes
    80:            OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.2.1
    92:            OCTET-STR (0x4) 3 bytes: "lo0"
    97:          SEQUENCE VARBIND (0x30): 18 bytes
    99:            OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.2.2
   111:            OCTET-STR (0x4) 4 bytes: "hme0"


interfaces.ifTable.ifEntry.ifIndex.1 : INTEGER: 1
interfaces.ifTable.ifEntry.ifIndex.2 : INTEGER: 2
interfaces.ifTable.ifEntry.ifIndex.3 : INTEGER: 3
interfaces.ifTable.ifEntry.ifDescr.1 : DISPLAY STRING- (ascii):  lo0
interfaces.ifTable.ifEntry.ifDescr.2 : DISPLAY STRING- (ascii):  hme0


 echo "TEST 10: HL: TRAP (snmpnotify) V2"
 echo ""
/opt/OV/bin/snmpnotify  -d -v 2c  bass .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"

TEST 10: HL: TRAP (snmpnotify) V2

Transmitted 108 bytes to bass (192.168.3.2) port 162:
     0:  30 6a 02 01 01 04 06 70 75 62 6c 69 63 a7 5d 02
0j.....public.].
    16:  02 6e 88 02 01 00 02 01 00 30 51 30 0d 06 08 2b
.n.......0Q0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 25 06 08 2b 06 01 02 01 01 01
.....0%..+......
    80:  00 04 19 53 75 6e 4f 53 20 65 63 6c 69 70 73 65     ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d -- -- -- --      5.3 1
sun4m....

     0:  SNMP MESSAGE (0x30): 106 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V2-TRAP-PDU (0xa7): 93 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 28296
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 81 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:           TIMETICKS (0x43) 1 bytes: 2
    42:         SEQUENCE VARBIND (0x30): 25 bytes
    44:           OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:           OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:         SEQUENCE VARBIND (0x30): 37 bytes
    71:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:           OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"
```

```
Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[306a020101040670756 26c6963a75d02026e880201000201003051300d06082b06010201
0103004301023019060a2b0601060301010401000 60b2b060104010b0211010003302506 0
82b060102010101000419537 56e4f532065636c6970736520352e332031207375 6e346d]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: V2_Trap
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[306a020101040670756 26c6963a75d02026e880201000201003051300d06082b06010201
0103004301023019060a2b0601060301010401000 60b2b060104010b0211010003302506 0
82b060102010101000419537 56e4f532065636c6970736520352e332031207375 6e346d]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 136 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 150
bytes.
```

```
 echo "TEST 11: HL: INFORMS (snmpnotify) V2"
 echo ""
/opt/OV/bin/snmpnotify  -d  -v 2c -A bass .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m "
```

```
Transmitted 109 bytes to bass (192.168.3.2) port 162:
Initial Timeout: 0.80 seconds
     0:  30 6b 02 01 01 04 06 70 75 62 6c 69 63 a6 5e 02
0k.....public.^.
    16:  02 10 ae 02 01 00 02 01 00 30 52 30 0d 06 08 2b
.........0R0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 26 06 08 2b 06 01 02 01 01 01
.....0&..+......
    80:  00 04 1a 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d 20 -- -- --       5.3 1 sun4m
...
```

```
     0:  SNMP MESSAGE (0x30): 107 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:     INFORM-REQUEST-PDU (0xa6): 94 bytes
    15:       INTEGER REQUEST-ID (0x2) 2 bytes: 4270
    19:       INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 82 bytes
    27:         SEQUENCE VARBIND (0x30): 13 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:           TIMETICKS (0x43) 1 bytes: 2
    42:         SEQUENCE VARBIND (0x30): 25 bytes
    44:           OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:           OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:         SEQUENCE VARBIND (0x30): 38 bytes
    71:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:           OCTET-STR (0x4) 26 bytes: "SunOS eclipse 5.3 1 sun4m "
```

```
Feb  6 13:45:46 myname high2low[317]: Entering UDP Case:
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall Packet is
[306b020101040670756 26c6963a65e020210ae020100020100305230 0d06082b06010201
0103000430102301906 0a2b06010603010104010006 0b2b060104010b021101000330 26060
82b0601020101010004 1a53756e4f532065636c697073652035 2e3320312073756e346d20
]
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname high2low[317]: Starting SNMP_PARSE: Inform
Feb  6 13:45:46 myname high2low[317]: DEBUG - Overall NEW Packet is
[306b020101040670756 26c6963a65e020210ae020100020100305230 0d06082b06010201
0103000430102301906 0a2b06010603010104010006 0b2b060104010b021101000330 26060
82b0601020101010004 1a53756e4f532065636c697073652035 2e3320312073756e346d20
]
Feb  6 13:45:46 myname high2low[317]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname high2low[317]: Status is 0
Feb  6 13:45:46 myname high2low[317]: 137 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- size_diff is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- change_made is 0
Feb  6 13:45:46 myname high2low[317]: DEBUG -- SendPacket(): writing 151
bytes.
```

```
Feb  6 13:45:46 myname low2high[319]: Entering UPD Case:
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Version
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: Community
String
Feb  6 13:45:46 myname low2high[319]: Starting SNMP_PARSE: GetResponse
Feb  6 13:45:46 myname low2high[319]: Entering STATUS Check on parse
return
Feb  6 13:45:46 myname low2high[319]: 137 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0
```

```
Received 109 bytes from bass (192.168.3.2) port 162:
     0:  30 6b 02 01 01 04 06 70 75 62 6c 69 63 a2 5e 02
0k.....public.^.
    16:  02 10 ae 02 01 00 02 01 00 30 52 30 0d 06 08 2b
.........0R0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 26 06 08 2b 06 01 02 01 01 01
.....0&..+......
    80:  00 04 1a 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d 20 -- -- --       5.3 1 sun4m
...

     0:  SNMP MESSAGE (0x30): 107 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 94 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 4270
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 82 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 2
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 38 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 26 bytes: "SunOS eclipse 5.3 1 sun4m "


Inform Object ID:
.iso.org.dod.internet.private.enterprises.hp.nm.openView.hpOpenView.0.3
system.sysDescr.0 : DISPLAY STRING- (ascii):  SunOS eclipse 5.3 1 sun4m
```

Example 2: NETMON Device configured with default recommended security policy. (See Table 1 for SNMP commands)

Note: HP Openview is configured to replay a request 3 times if there is no response received. Therefore, when the NETMON device is configured to disallow a command the output from the high-side NMS will show the request 3 times.

KEY: Yellow High-Side NMS
Blue is response from low-side NMS as seen on high-side NMS
Green is an allowed at the NETMON device (no details)
Red is a disallowed at the NETMON device (no details)

```
 echo "TEST 1: HL: SETREQUEST V1"
 echo ""
/opt/OV/bin/snmpset -v 1 -d bass system.sysContact.0 octetstring "BOB
JONES"

TEST 1: HL: SETREQUEST V1

Transmitted 50 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a3 23 02
00.....netmon.#.
    16:  02 7e 18 02 01 00 02 01 00 30 17 30 15 06 08 2b
.~.......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............

     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 32280
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

Allowed through proxy.

Received 50 bytes from bass (192.168.3.2) port 161:
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a2 23 02
00.....netmon.#.
    16:  02 7e 18 02 01 00 02 01 00 30 17 30 15 06 08 2b
.~.......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............
```

```
     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    RESPONSE-PDU (0xa2): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 32280
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

system.sysContact.0 : DISPLAY STRING- (ascii):  BOB JONES



 echo "TEST 2: HL: GETREQUEST V1"
 echo ""
/opt/OV/bin/snmpget -v 1 -d bass system.sysContact.0

TEST 2: HL: GETREQUEST V1

Transmitted 41 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 00 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 7b 4c 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.{L......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 31564
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes

Allowed through proxy.

Received 50 bytes from bass (192.168.3.2) port 161:
     0:  30 30 02 01 00 04 06 70 75 62 6c 69 63 a2 23 02
00.....public.#.
    16:  02 7b 4c 02 01 00 02 01 00 30 17 30 15 06 08 2b
.{L......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............
```

```
     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 31564
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

system.sysContact.0 : DISPLAY STRING- (ascii):  BOB JONES
```

```
 echo "TEST 3: HL: GETNEXTREQUEST V1"
 echo ""
/opt/OV/bin/snmpnext -v 1 -d bass interfaces.ifTable.ifEntry.0
```

```
TEST 3: HL: GETNEXTREQUEST V1

Transmitted 42 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 28 02 01 00 04 06 70 75 62 6c 69 63 a1 1b 02
0(.....public...
    16:  02 4a 41 02 01 00 02 01 00 30 0f 30 0d 06 09 2b
.JA......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 19009
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Allowed through proxy.

Received 44 bytes from bass (192.168.3.2) port 161:
     0:  30 2a 02 01 00 04 06 70 75 62 6c 69 63 a2 1d 02
0*.....public...
    16:  02 4a 41 02 01 00 02 01 00 30 11 30 0f 06 0a 2b
.JA......0.0...+
    32:  06 01 02 01 02 02 01 01 01 02 01 01 -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 42 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 29 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 19009
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 17 bytes
    27:        SEQUENCE VARBIND (0x30): 15 bytes
    29:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.1
    41:          INTEGER (0x2) 1 bytes: 1


interfaces.ifTable.ifEntry.ifIndex.1 : INTEGER: 1



 echo "TEST 4: HL: TRAP (snmptrap) V1"
 echo ""
/opt/OV/bin/snmptrap -d bass "" "" 6 1 15 system.sysDescr.0
octetstringascii "SunOS hpcndnw2 4.1 1 sun4c"

TEST 4: HL: TRAP (snmptrap) V1

Transmitted 85 bytes to bass (192.168.3.2) port 162:
     0:  30 53 02 01 00 04 06 70 75 62 6c 69 63 a4 46 06
0S.....public.F.
    16:  0b 2b 06 01 04 01 0b 02 03 0a 01 02 40 04 c0 a8
.+..........@...
    32:  01 01 02 01 06 02 01 01 43 01 0f 30 28 30 26 06
........C..0(0&.
    48:  08 2b 06 01 02 01 01 01 00 04 1a 53 75 6e 4f 53
.+.........SunOS
    64:  20 68 70 63 6e 64 6e 77 32 20 34 2e 31 20 31 20      hpcndnw2
4.1 1
    80:  73 75 6e 34 63 -- -- -- -- -- -- -- -- -- -- --
sun4c...........

     0:  SNMP MESSAGE (0x30): 83 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 70 bytes
    15:      OBJ-ID ENTERPRISE (0x6) 11 bytes: .1.3.6.1.4.1.11.2.3.10.1.2
    28:      IPADDRESS AGENT-ADDR (0x40) 4 bytes: 192.168.1.1
(COPhost.ndf.af.mil)
    34:      INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    37:      INTEGER SPECIFIC-TRAP (0x2) 1 bytes: 1
    40:      TIMETICKS TIME-STAMP (0x43) 1 bytes: 15 (0xf)
    43:      SEQUENCE VARBIND-LIST (0x30): 40 bytes
    45:        SEQUENCE VARBIND (0x30): 38 bytes
    47:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    57:          OCTET-STR (0x4) 26 bytes: "SunOS hpcndnw2 4.1 1 sun4c"


Denied at Proxy.

 echo "TEST 5: HL: TRAP (snmpnotify) V1"
 echo ""
```

```
/opt/OV/bin/snmpnotify  -d  -v 1 bass .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"

TEST 5: HL: TRAP (snmpnotify) V1

Transmitted 82 bytes to bass (192.168.3.2) port 162:
     0:  30 50 02 01 00 04 06 70 75 62 6c 69 63 a4 43 06
0P.....public.C.
    16:  09 2b 06 01 04 01 0b 02 11 01 40 04 c0 a8 01 01
.+........@.....
    32:  02 01 06 02 01 03 43 01 02 30 27 30 25 06 08 2b
......C..0'0%..+
    48:  06 01 02 01 01 01 00 04 19 53 75 6e 4f 53 20 65
.........SunOS e
    64:  63 6c 69 70 73 65 20 35 2e 33 20 31 20 73 75 6e     clipse 5.3 1
sun
    80:  34 6d -- -- -- -- -- -- -- -- -- -- -- -- -- --
4m..............

     0:  SNMP MESSAGE (0x30): 80 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 67 bytes
    15:      OBJ-ID ENTERPRISE (0x6) 9 bytes: .1.3.6.1.4.1.11.2.17.1
    26:      IPADDRESS AGENT-ADDR (0x40) 4 bytes: 192.168.1.1
(COPhost.ndf.af.mil)
    32:      INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    35:      INTEGER SPECIFIC-TRAP (0x2) 1 bytes: 3
    38:      TIMETICKS TIME-STAMP (0x43) 1 bytes: 2 (0x2)
    41:      SEQUENCE VARBIND-LIST (0x30): 39 bytes
    43:        SEQUENCE VARBIND (0x30): 37 bytes
    45:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    55:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"


Denied at Proxy.

echo "TEST 6: HL: SETREQUEST V2"
 echo ""
/opt/OV/bin/snmpset -v 2c -d bass system.sysContact.0 octetstring "Herb
Markle "

TEST 6: HL: SETREQUEST V2

Transmitted 53 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a3 26 02
03.....netmon.&.
    16:  02 59 5a 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.YZ......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 22874
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "

Allowed through proxy.

Received 53 bytes from bass (192.168.3.2) port 161:
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a2 26 02
03.....netmon.&.
    16:  02 59 5a 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.YZ......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    RESPONSE-PDU (0xa2): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 22874
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "


system.sysContact.0 : DISPLAY STRING- (ascii):  Herb Markle


 echo "TEST 7: HL: GETREQUEST V2"
 echo ""
/opt/OV/bin/snmpget -v 2c -d bass system.sysContact.0

TEST 7: HL: GETREQUEST V2

Transmitted 41 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 2f 21 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
./!......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:       INTEGER REQUEST-ID (0x2) 2 bytes: 12065
    19:       INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:         SEQUENCE VARBIND (0x30): 12 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:           NULL (0x5) 0 bytes

Allowed through proxy.

Received 53 bytes from bass (192.168.3.2) port 161:
     0:  30 33 02 01 01 04 06 70 75 62 6c 69 63 a2 26 02
03.....public.&.
    16:  02 2f 21 02 01 00 02 01 00 30 1a 30 18 06 08 2b
./!......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 12065
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "


system.sysContact.0 : DISPLAY STRING- (ascii):  Herb Markle


 echo "TEST 8: HL: GETNEXTREQUEST V2"
 echo ""
/opt/OV/bin/snmpnext -v 2c -d  bass system.sysContact.1

TEST 8: HL: GETNEXTREQUEST V2

Transmitted 41 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a1 1a 02
0'.....public...
    16:  02 5c 0d 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.\.......0.0...+
    32:  06 01 02 01 01 04 01 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 23565
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:         SEQUENCE VARBIND (0x30): 12 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.1
    39:           NULL (0x5) 0 bytes

Allowed through proxy.

Received 45 bytes from bass (192.168.3.2) port 161:
     0:  30 2b 02 01 01 04 06 70 75 62 6c 69 63 a2 1e 02
0+.....public...
    16:  02 5c 0d 02 01 00 02 01 00 30 12 30 10 06 08 2b
.\.......0.0...+
    32:  06 01 02 01 01 05 00 04 04 62 61 73 73 -- -- --
.........bass...

     0:  SNMP MESSAGE (0x30): 43 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 30 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 23565
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 18 bytes
    27:        SEQUENCE VARBIND (0x30): 16 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.5.0
    39:          OCTET-STR (0x4) 4 bytes: "bass"


system.sysName.0 : DISPLAY STRING- (ascii):  bass

 echo "TEST 9: HL: GETBULKREQUEST V2"
 echo ""
/opt/OV/bin/snmpbulk -v2c -d  bass interfaces.ifTable.ifEntry.0

TEST 9: HL: GETBULKREQUEST V2

Transmitted 42 bytes to bass (192.168.3.2) port 161:
Initial Timeout: 0.80 seconds
     0:  30 28 02 01 01 04 06 70 75 62 6c 69 63 a5 1b 02
0(.....public...
    16:  02 0b 2a 02 01 00 02 01 05 30 0f 30 0d 06 09 2b
..*......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETBULK-REQUEST-PDU (0xa5): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 2858
    19:      INTEGER NON-REPEATERS (0x2) 1 bytes: 0
    22:      INTEGER MAX-REPETITIONS (0x2) 1 bytes: 5
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes
```

```
Allowed through proxy.

Received 117 bytes from bass (192.168.3.2) port 161:
     0:  30 73 02 01 01 04 06 70 75 62 6c 69 63 a2 66 02
0s.....public.f.
    16:  02 0b 2a 02 01 00 02 01 00 30 5a 30 0f 06 0a 2b
..*......0Z0...+
    32:  06 01 02 01 02 02 01 01 01 02 01 01 30 0f 06 0a
............0...
    48:  2b 06 01 02 01 02 02 01 01 02 02 01 02 30 0f 06
+............0..
    64:  0a 2b 06 01 02 01 02 02 01 01 03 02 01 03 30 11
.+............0.
    80:  06 0a 2b 06 01 02 01 02 02 01 02 01 04 03 6c 6f
..+...........lo
    96:  30 30 12 06 0a 2b 06 01 02 01 02 02 01 02 02 04
00...+..........
   112:  04 68 6d 65 30 -- -- -- -- -- -- -- -- -- -- --
.hme0...........

     0:  SNMP MESSAGE (0x30): 115 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    RESPONSE-PDU (0xa2): 102 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 2858
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 90 bytes
    27:        SEQUENCE VARBIND (0x30): 15 bytes
    29:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.1
    41:          INTEGER (0x2) 1 bytes: 1
    44:        SEQUENCE VARBIND (0x30): 15 bytes
    46:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.2
    58:          INTEGER (0x2) 1 bytes: 2
    61:        SEQUENCE VARBIND (0x30): 15 bytes
    63:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.1.3
    75:          INTEGER (0x2) 1 bytes: 3
    78:        SEQUENCE VARBIND (0x30): 17 bytes
    80:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.2.1
    92:          OCTET-STR (0x4) 3 bytes: "lo0"
    97:        SEQUENCE VARBIND (0x30): 18 bytes
    99:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.2.1.2.2.1.2.2
   111:          OCTET-STR (0x4) 4 bytes: "hme0"


interfaces.ifTable.ifEntry.ifIndex.1 : INTEGER: 1
interfaces.ifTable.ifEntry.ifIndex.2 : INTEGER: 2
interfaces.ifTable.ifEntry.ifIndex.3 : INTEGER: 3
interfaces.ifTable.ifEntry.ifDescr.1 : DISPLAY STRING- (ascii):  lo0
interfaces.ifTable.ifEntry.ifDescr.2 : DISPLAY STRING- (ascii):  hme0


 echo "TEST 10: HL: TRAP (snmpnotify) V2"
 echo ""
/opt/OV/bin/snmpnotify  -d -v 2c  bass .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"
```

```
TEST 10: HL: TRAP (snmpnotify) V2

Transmitted 108 bytes to bass (192.168.3.2) port 162:
     0:  30 6a 02 01 01 04 06 70 75 62 6c 69 63 a7 5d 02
0j.....public.].
    16:  02 5e fe 02 01 00 02 01 00 30 51 30 0d 06 08 2b
.^.......0Q0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 25 06 08 2b 06 01 02 01 01 01
.....0%..+......
    80:  00 04 19 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d -- -- -- --       5.3 1
sun4m....

     0:  SNMP MESSAGE (0x30): 106 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V2-TRAP-PDU (0xa7): 93 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 24318
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 81 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 2
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 37 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"

Denied at Proxy.

 echo "TEST 11: HL: INFORMS (snmpnotify) V2"
 echo ""
/opt/OV/bin/snmpnotify  -d  -v 2c -A bass .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m "

TEST 11: HL: INFORMS (snmpnotify) V2

Transmitted 109 bytes to bass (192.168.3.2) port 162:
Initial Timeout: 0.80 seconds
     0:  30 6b 02 01 01 04 06 70 75 62 6c 69 63 a6 5e 02
0k.....public.^.
    16:  02 66 16 02 01 00 02 01 00 30 52 30 0d 06 08 2b
.f.......0R0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
```

```
    64:  02 11 01 00 03 30 26 06 08 2b 06 01 02 01 01 01
.....0&..+......
    80:  00 04 1a 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d 20 -- -- --       5.3 1 sun4m
...

     0:  SNMP MESSAGE (0x30): 107 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    INFORM-REQUEST-PDU (0xa6): 94 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 26134
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 82 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 2
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 38 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 26 bytes: "SunOS eclipse 5.3 1 sun4m "

Denied at Proxy.

Transmitted 109 bytes to bass (192.168.3.2) port 162:
Retry #1  Timeout: 1.60 seconds
     0:  30 6b 02 01 01 04 06 70 75 62 6c 69 63 a6 5e 02
0k.....public.^.
    16:  02 66 16 02 01 00 02 01 00 30 52 30 0d 06 08 2b
.f.......0R0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 26 06 08 2b 06 01 02 01 01 01
.....0&..+......
    80:  00 04 1a 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d 20 -- -- --       5.3 1 sun4m
...

     0:  SNMP MESSAGE (0x30): 107 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    INFORM-REQUEST-PDU (0xa6): 94 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 26134
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 82 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 2
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 38 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 26 bytes: "SunOS eclipse 5.3 1 sun4m "

Denied at Proxy.

Transmitted 109 bytes to bass (192.168.3.2) port 162:
Retry #2  Timeout: 3.20 seconds
     0:  30 6b 02 01 01 04 06 70 75 62 6c 69 63 a6 5e 02
0k.....public.^.
    16:  02 66 16 02 01 00 02 01 00 30 52 30 0d 06 08 2b
.f.......0R0...+
    32:  06 01 02 01 01 03 00 43 01 02 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 26 06 08 2b 06 01 02 01 01 01
.....0&..+......
    80:  00 04 1a 53 75 6e 4f 53 20 65 63 6c 69 70 73 65     ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d 20 -- -- --      5.3 1 sun4m
...

     0:  SNMP MESSAGE (0x30): 107 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    INFORM-REQUEST-PDU (0xa6): 94 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 26134
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 82 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 2
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 38 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 26 bytes: "SunOS eclipse 5.3 1 sun4m "

Denied at Proxy.
```

Example 3: NETMON Device configured with default recommended security policy. (See Table 1 for SNMP commands)

Note: HP Openview is configured to replay a request 3 times if there is no response received. Therefore, when the NETMON device is configured to disallow a command the output from the low-side NMS will show the request 3 times.

KEY: Yellow Low-Side NMS
Blue is response from high-side NMS as seen on low-side NMS
Green is an allowed at the NETMON device (no details)
Red is a disallowed at the NETMON device (no details)

```
echo "TEST 1: LH: SETREQUEST V1"
 echo ""
$OV_BIN/snmpset -v 1 -c netmon -d 192.168.3.1 system.sysContact.0
octetstring "BOB JONES"


TEST 1: LH: SETREQUEST V1

Transmitted 50 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a3 23 02
00.....netmon.#.
    16:  02 35 d0 02 01 00 02 01 00 30 17 30 15 06 08 2b
.5.......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............

     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 13776
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

Denied at Proxy.

Transmitted 50 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a3 23 02
00.....netmon.#.
    16:  02 35 d0 02 01 00 02 01 00 30 17 30 15 06 08 2b
.5.......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
```

```
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............

     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 13776
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

Denied at Proxy.

Transmitted 50 bytes to herb (192.168.3.1) port 161:
Retry #2  Timeout: 3.20 seconds
     0:  30 30 02 01 00 04 06 6e 65 74 6d 6f 6e a3 23 02
00.....netmon.#.
    16:  02 35 d0 02 01 00 02 01 00 30 17 30 15 06 08 2b
.5.......0.0...+
    32:  06 01 02 01 01 04 00 04 09 42 4f 42 20 4a 4f 4e      .........BOB
JON
    48:  45 53 -- -- -- -- -- -- -- -- -- -- -- -- -- --
ES..............

     0:  SNMP MESSAGE (0x30): 48 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 35 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 13776
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 23 bytes
    27:        SEQUENCE VARBIND (0x30): 21 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 9 bytes: "BOB JONES"

Denied at Proxy.


echo "TEST 2: LH: GETREQUEST V1"
 echo ""
$OV_BIN/snmpget -v 1 -d 192.168.3.1 system.sysContact.0

TEST 2: LH: GETREQUEST V1

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 00 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 47 d3 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.G.......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................
```

```
     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 18387
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 27 02 01 00 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 47 d3 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.G.......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 18387
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Retry #2  Timeout: 3.20 seconds
     0:  30 27 02 01 00 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 47 d3 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.G.......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 18387
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes

Denied at Proxy.

echo "TEST 3: LH: GETNEXTREQUEST V1"
 echo ""
$OV_BIN/snmpnext -v 1 -d 192.168.3.1 interfaces.ifTable.ifEntry.0

TEST 3: LH: GETNEXTREQUEST V1

Transmitted 42 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:  30 28 02 01 00 04 06 70 75 62 6c 69 63 a1 1b 02
0(.....public...
    16:  02 3f e2 02 01 00 02 01 00 30 0f 30 0d 06 09 2b
.?.......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 16354
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 42 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 28 02 01 00 04 06 70 75 62 6c 69 63 a1 1b 02
0(.....public...
    16:  02 3f e2 02 01 00 02 01 00 30 0f 30 0d 06 09 2b
.?.......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 16354
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 42 bytes to herb (192.168.3.1) port 161:
```

```
Retry #2  Timeout: 3.20 seconds
     0:  30 28 02 01 00 04 06 70 75 62 6c 69 63 a1 1b 02
0(.....public...
    16:  02 3f e2 02 01 00 02 01 00 30 0f 30 0d 06 09 2b
.?.......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 16354
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Denied at Proxy.

echo "TEST 4: LH: TRAP (snmptrap) V1"
 echo ""
$OV_BIN/snmptrap -d 192.168.3.1 "" "" 6 1 15 system.sysDescr.0
octetstringascii "SunOS hpcndnw2 4.1 1 sun4c"

TEST 4: LH: TRAP (snmptrap) V1

Transmitted 85 bytes to herb (192.168.3.1) port 162:
     0:  30 53 02 01 00 04 06 70 75 62 6c 69 63 a4 46 06
0S.....public.F.
    16:  0b 2b 06 01 04 01 0b 02 03 0a 01 02 40 04 c0 a8
.+..........@...
    32:  03 02 02 01 06 02 01 01 43 01 0f 30 28 30 26 06
........C..0(0&.
    48:  08 2b 06 01 02 01 01 01 00 04 1a 53 75 6e 4f 53
.+.........SunOS
    64:  20 68 70 63 6e 64 6e 77 32 20 34 2e 31 20 31 20      hpcndnw2
4.1 1
    80:  73 75 6e 34 63 -- -- -- -- -- -- -- -- -- -- --
sun4c...........

     0:  SNMP MESSAGE (0x30): 83 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 70 bytes
    15:      OBJ-ID ENTERPRISE (0x6) 11 bytes: .1.3.6.1.4.1.11.2.3.10.1.2
    28:      IPADDRESS AGENT-ADDR (0x40) 4 bytes: 192.168.3.2 (bass)
    34:      INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    37:      INTEGER SPECIFIC-TRAP (0x2) 1 bytes: 1
    40:      TIMETICKS TIME-STAMP (0x43) 1 bytes: 15 (0xf)
    43:      SEQUENCE VARBIND-LIST (0x30): 40 bytes
    45:        SEQUENCE VARBIND (0x30): 38 bytes
    47:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    57:          OCTET-STR (0x4) 26 bytes: "SunOS hpcndnw2 4.1 1 sun4c"
```

```
Allowed through proxy.

echo "TEST 5: LH: TRAP (snmpnotify) V1"
 echo ""
$OV_BIN/snmpnotify  -d  -v 1 herb .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"


TEST 5: LH: TRAP (snmpnotify) V1

Transmitted 82 bytes to herb (192.168.3.1) port 162:
     0:  30 50 02 01 00 04 06 70 75 62 6c 69 63 a4 43 06
0P.....public.C.
    16:  09 2b 06 01 04 01 0b 02 11 01 40 04 c0 a8 03 02
.+........@.....
    32:  02 01 06 02 01 03 43 01 01 30 27 30 25 06 08 2b
......C..0'0%..+
    48:  06 01 02 01 01 01 00 04 19 53 75 6e 4f 53 20 65
.........SunOS e
    64:  63 6c 69 70 73 65 20 35 2e 33 20 31 20 73 75 6e     clipse 5.3 1
sun
    80:  34 6d -- -- -- -- -- -- -- -- -- -- -- -- -- --
4m..............

     0:  SNMP MESSAGE (0x30): 80 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 0 (SNMPv1)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V1-TRAP-PDU (0xa4): 67 bytes
    15:      OBJ-ID ENTERPRISE (0x6) 9 bytes: .1.3.6.1.4.1.11.2.17.1
    26:      IPADDRESS AGENT-ADDR (0x40) 4 bytes: 192.168.3.2 (bass)
    32:      INTEGER GENERIC-TRAP (0x2) 1 bytes: 6
    35:      INTEGER SPECIFIC-TRAP (0x2) 1 bytes: 3
    38:      TIMETICKS TIME-STAMP (0x43) 1 bytes: 1 (0x1)
    41:      SEQUENCE VARBIND-LIST (0x30): 39 bytes
    43:        SEQUENCE VARBIND (0x30): 37 bytes
    45:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    55:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"

Allowed through proxy.

echo "TEST 6: LH: SETREQUEST V2"
 echo ""
$OV_BIN/snmpset -v 2c -c netmon -d 192.168.3.1 system.sysContact.0
octetstring "Herb Markle "

TEST 6: LH: SETREQUEST V2

Transmitted 53 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a3 26 02
03.....netmon.&.
    16:  02 21 7a 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.!z......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........
```

```
     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 8570
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "

Denied at Proxy.

Transmitted 53 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a3 26 02
03.....netmon.&.
    16:  02 21 7a 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.!z......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 38 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 8570
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:        SEQUENCE VARBIND (0x30): 24 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          OCTET-STR (0x4) 12 bytes: "Herb Markle "


Transmitted 53 bytes to herb (192.168.3.1) port 161:

Denied at Proxy.

Retry #2  Timeout: 3.20 seconds
     0:  30 33 02 01 01 04 06 6e 65 74 6d 6f 6e a3 26 02
03.....netmon.&.
    16:  02 21 7a 02 01 00 02 01 00 30 1a 30 18 06 08 2b
.!z......0.0...+
    32:  06 01 02 01 01 04 00 04 0c 48 65 72 62 20 4d 61
.........Herb Ma
    48:  72 6b 6c 65 20 -- -- -- -- -- -- -- -- -- -- --     rkle
...........

     0:  SNMP MESSAGE (0x30): 51 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "netmon"
    13:    SET-REQUEST-PDU (0xa3): 38 bytes
    15:       INTEGER REQUEST-ID (0x2) 2 bytes: 8570
    19:       INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 26 bytes
    27:         SEQUENCE VARBIND (0x30): 24 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:           OCTET-STR (0x4) 12 bytes: "Herb Markle "

Denied at Proxy.

echo "TEST 7: LH: GETREQUEST V2"
 echo ""
$OV_BIN/snmpget -v 2c -d 192.168.3.1 system.sysContact.0

TEST 7: LH: GETREQUEST V2

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 78 6e 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.xn......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 30830
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:  02 78 6e 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.xn......0.0...+
    32:  06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GET-REQUEST-PDU (0xa0): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 30830
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:            OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:            NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Retry #2  Timeout: 3.20 seconds
     0:   30 27 02 01 01 04 06 70 75 62 6c 69 63 a0 1a 02
0'.....public...
    16:   02 78 6e 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.xn......0.0...+
    32:   06 01 02 01 01 04 00 05 00 -- -- -- -- -- -- --
................

     0:   SNMP MESSAGE (0x30): 39 bytes
     2:     INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:     OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:     GET-REQUEST-PDU (0xa0): 26 bytes
    15:       INTEGER REQUEST-ID (0x2) 2 bytes: 30830
    19:       INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:         SEQUENCE VARBIND (0x30): 12 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.0
    39:           NULL (0x5) 0 bytes

Denied at Proxy.

echo "TEST 8: LH: GETNEXTREQUEST V2"
 echo ""
$OV_BIN/snmpnext -v 2c -d  192.168.3.1 system.sysContact.1

TEST 8: LH: GETNEXTREQUEST V2

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:   30 27 02 01 01 04 06 70 75 62 6c 69 63 a1 1a 02
0'.....public...
    16:   02 43 29 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.C)......0.0...+
    32:   06 01 02 01 01 04 01 05 00 -- -- -- -- -- -- --
................

     0:   SNMP MESSAGE (0x30): 39 bytes
     2:     INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:     OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:     GETNEXT-REQUEST-PDU (0xa1): 26 bytes
    15:       INTEGER REQUEST-ID (0x2) 2 bytes: 17193
    19:       INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:         SEQUENCE VARBIND (0x30): 12 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.1
    39:           NULL (0x5) 0 bytes

Denied at Proxy.
```

```
Transmitted 41 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a1 1a 02
0'.....public...
    16:  02 43 29 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.C)......0.0...+
    32:  06 01 02 01 01 04 01 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 17193
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.1
    39:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 41 bytes to herb (192.168.3.1) port 161:
Retry #2  Timeout: 3.20 seconds
     0:  30 27 02 01 01 04 06 70 75 62 6c 69 63 a1 1a 02
0'.....public...
    16:  02 43 29 02 01 00 02 01 00 30 0e 30 0c 06 08 2b
.C)......0.0...+
    32:  06 01 02 01 01 04 01 05 00 -- -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 39 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETNEXT-REQUEST-PDU (0xa1): 26 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 17193
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 14 bytes
    27:        SEQUENCE VARBIND (0x30): 12 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.4.1
    39:          NULL (0x5) 0 bytes
Denied at Proxy.

echo "TEST 9: LH: GETBULKREQUEST V2"
 echo ""
$OV_BIN/snmpbulk -v2c -d  192.168.3.1 interfaces.ifTable.ifEntry.0

TEST 9: LH: GETBULKREQUEST V2

Transmitted 42 bytes to herb (192.168.3.1) port 161:
Initial Timeout: 0.80 seconds
     0:  30 28 02 01 01 04 06 70 75 62 6c 69 63 a5 1b 02
0(.....public...
    16:  02 24 2a 02 01 00 02 01 05 30 0f 30 0d 06 09 2b
.$*......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETBULK-REQUEST-PDU (0xa5): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9258
    19:      INTEGER NON-REPEATERS (0x2) 1 bytes: 0
    22:      INTEGER MAX-REPETITIONS (0x2) 1 bytes: 5
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 42 bytes to herb (192.168.3.1) port 161:
Retry #1  Timeout: 1.60 seconds
     0:  30 28 02 01 01 04 06 70 75 62 6c 69 63 a5 1b 02
0(.....public...
    16:  02 24 2a 02 01 00 02 01 05 30 0f 30 0d 06 09 2b
.$*......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETBULK-REQUEST-PDU (0xa5): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9258
    19:      INTEGER NON-REPEATERS (0x2) 1 bytes: 0
    22:      INTEGER MAX-REPETITIONS (0x2) 1 bytes: 5
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:          NULL (0x5) 0 bytes

Denied at Proxy.

Transmitted 42 bytes to herb (192.168.3.1) port 161:
Retry #2  Timeout: 3.20 seconds
     0:  30 28 02 01 01 04 06 70 75 62 6c 69 63 a5 1b 02
0(.....public...
    16:  02 24 2a 02 01 00 02 01 05 30 0f 30 0d 06 09 2b
.$*......0.0...+
    32:  06 01 02 01 02 02 01 00 05 00 -- -- -- -- -- --
................

     0:  SNMP MESSAGE (0x30): 40 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    GETBULK-REQUEST-PDU (0xa5): 27 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 9258
    19:      INTEGER NON-REPEATERS (0x2) 1 bytes: 0
    22:      INTEGER MAX-REPETITIONS (0x2) 1 bytes: 5
    25:      SEQUENCE VARBIND-LIST (0x30): 15 bytes
    27:         SEQUENCE VARBIND (0x30): 13 bytes
    29:           OBJ-ID (0x6) 9 bytes: .1.3.6.1.2.1.2.2.1.0
    40:           NULL (0x5) 0 bytes
```

Denied at Proxy.

```
 echo "TEST 10: LH: TRAP (snmpnotify) V2"
 echo ""
$OV_BIN/snmpnotify  -d -v 2c 192.168.3.1  .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"
```

```
TEST 10: LH: TRAP (snmpnotify) V2

Transmitted 108 bytes to herb (192.168.3.1) port 162:
     0:  30 6a 02 01 01 04 06 70 75 62 6c 69 63 a7 5d 02
0j.....public.].
    16:  02 3a d8 02 01 00 02 01 00 30 51 30 0d 06 08 2b
.:.......0Q0...+
    32:  06 01 02 01 01 03 00 43 01 01 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 25 06 08 2b 06 01 02 01 01 01
.....0%..+......
    80:  00 04 19 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d -- -- -- --       5.3 1
sun4m....

     0:  SNMP MESSAGE (0x30): 106 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    V2-TRAP-PDU (0xa7): 93 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 15064
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 81 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 1
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 37 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"
```

Allowed through proxy.

```
echo "TEST 11: LH: INFORMS (snmpnotify) V2"
 echo ""
$OV_BIN/snmpnotify  -d -v 2c -A herb  .1.3.6.1.4.1.11.2.17.1.0.3
system.sysDescr.0 octetstringascii "SunOS eclipse 5.3 1 sun4m"
```

```
TEST 11: LH: INFORMS (snmpnotify) V2

Transmitted 108 bytes to herb (192.168.3.1) port 162:
```

```
Initial Timeout: 0.80 seconds
     0:  30 6a 02 01 01 04 06 70 75 62 6c 69 63 a6 5d 02
0j.....public.].
    16:  02 14 de 02 01 00 02 01 00 30 51 30 0d 06 08 2b
.........0Q0...+
    32:  06 01 02 01 01 03 00 43 01 01 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 25 06 08 2b 06 01 02 01 01 01
.....0%..+......
    80:  00 04 19 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d -- -- -- --       5.3 1
sun4m....

     0:  SNMP MESSAGE (0x30): 106 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    INFORM-REQUEST-PDU (0xa6): 93 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 5342
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 81 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 1
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 37 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"

Denied at Proxy.

Transmitted 108 bytes to herb (192.168.3.1) port 162:
Retry #1  Timeout: 1.60 seconds
     0:  30 6a 02 01 01 04 06 70 75 62 6c 69 63 a6 5d 02
0j.....public.].
    16:  02 14 de 02 01 00 02 01 00 30 51 30 0d 06 08 2b
.........0Q0...+
    32:  06 01 02 01 01 03 00 43 01 01 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 25 06 08 2b 06 01 02 01 01 01
.....0%..+......
    80:  00 04 19 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d -- -- -- --       5.3 1
sun4m....

     0:  SNMP MESSAGE (0x30): 106 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    INFORM-REQUEST-PDU (0xa6): 93 bytes
    15:       INTEGER REQUEST-ID (0x2) 2 bytes: 5342
    19:       INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:       INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:       SEQUENCE VARBIND-LIST (0x30): 81 bytes
    27:         SEQUENCE VARBIND (0x30): 13 bytes
    29:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:           TIMETICKS (0x43) 1 bytes: 1
    42:         SEQUENCE VARBIND (0x30): 25 bytes
    44:           OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:           OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:         SEQUENCE VARBIND (0x30): 37 bytes
    71:           OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:           OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"
```

```
Denied at Proxy.
```

```
Transmitted 108 bytes to herb (192.168.3.1) port 162:
Retry #2  Timeout: 3.20 seconds
     0:  30 6a 02 01 01 04 06 70 75 62 6c 69 63 a6 5d 02
0j.....public.].
    16:  02 14 de 02 01 00 02 01 00 30 51 30 0d 06 08 2b
.........0Q0...+
    32:  06 01 02 01 01 03 00 43 01 01 30 19 06 0a 2b 06
.......C..0...+.
    48:  01 06 03 01 01 04 01 00 06 0b 2b 06 01 04 01 0b
..........+.....
    64:  02 11 01 00 03 30 25 06 08 2b 06 01 02 01 01 01
.....0%..+......
    80:  00 04 19 53 75 6e 4f 53 20 65 63 6c 69 70 73 65      ...SunOS
eclipse
    96:  20 35 2e 33 20 31 20 73 75 6e 34 6d -- -- -- --       5.3 1
sun4m....
```

```
     0:  SNMP MESSAGE (0x30): 106 bytes
     2:    INTEGER VERSION (0x2) 1 bytes: 1 (SNMPv2C)
     5:    OCTET-STR COMMUNITY (0x4) 6 bytes: "public"
    13:    INFORM-REQUEST-PDU (0xa6): 93 bytes
    15:      INTEGER REQUEST-ID (0x2) 2 bytes: 5342
    19:      INTEGER ERROR-STATUS (0x2) 1 bytes: noError(0)
    22:      INTEGER ERROR-INDEX (0x2) 1 bytes: 0
    25:      SEQUENCE VARBIND-LIST (0x30): 81 bytes
    27:        SEQUENCE VARBIND (0x30): 13 bytes
    29:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.3.0
    39:          TIMETICKS (0x43) 1 bytes: 1
    42:        SEQUENCE VARBIND (0x30): 25 bytes
    44:          OBJ-ID (0x6) 10 bytes: .1.3.6.1.6.3.1.1.4.1.0
    56:          OBJ-ID (0x6) 11 bytes: .1.3.6.1.4.1.11.2.17.1.0.3
    69:        SEQUENCE VARBIND (0x30): 37 bytes
    71:          OBJ-ID (0x6) 8 bytes: .1.3.6.1.2.1.1.1.0
    81:          OCTET-STR (0x4) 25 bytes: "SunOS eclipse 5.3 1 sun4m"
```

Denied at Proxy.

Example 4: Audit trail of NETMON device for a high-side Getrequest and the Response from the low-side NMS.

KEY: Yellow high-side input to device
Blue low-side input to device
Green is showing data field sanitization of high-side NMS hostname

```
Feb  7 17:11:27 myname high2low[2605]: Entering UDP Case:
Feb  7 17:11:27 myname high2low[2605]: DEBUG - Overall Packet is
[302c02010104066e65746d6f6ea31f02024de902010002010030133011060 82b06010201
01040004056861727073]
Feb  7 17:11:27 myname high2low[2605]: Starting SNMP_PARSE: Version
Feb  7 17:11:27 myname high2low[2605]: Starting SNMP_PARSE: Community
String
Feb  7 17:11:27 myname high2low[2605]: Starting SNMP_PARSE: SetRequest
Feb  7 17:11:27 myname high2low[2605]:  SETREQ High to Low:
02024de90201000201003013301106082b0601020101040004056861727073 2
Feb  7 17:11:27 myname high2low[2605]: DEBUG - Entering snmp_pdu_parse()
Feb  7 17:11:27 myname high2low[2605]: DEBUG - Entering
snmp_varbind_parse()
Feb  7 17:11:27 myname high2low[2605]: DEBUG - count is [10]
Feb  7 17:11:27 myname high2low[2605]: DEBUG - len_change is [0]
Feb  7 17:11:27 myname high2low[2605]: Error is 0, ind is 1, and
elem.type is 6
Feb  7 17:11:27 myname high2low[2605]: DEBUG - Looking for harps
Feb  7 17:11:27 myname high2low[2605]: DEBUG - Found harps, replacing
with herbs.domain.long.name.extra
Feb  7 17:11:27 myname high2low[2605]: Adjusted ASN1 length field for
hostname from 5 to 28
Feb  7 17:11:27 myname high2low[2605]: DEBUG -old_size is 31 (0x1F),
new_size is 54 (0x36)
Feb  7 17:11:27 myname high2low[2605]: DEBUG -old_size is 44 (0x2C),
new_size is 67 (0x43)
Feb  7 17:11:27 myname high2low[2605]: DEBUG - Overall NEW Packet is
[304302010104066e65746d6f6ea33602024de902010002010030 2a302806082b06010201
010400041c68657262732e646f6d61696e2e6c6f6e672e6e616d652e6578747261]
Feb  7 17:11:27 myname high2low[2605]: Entering STATUS Check on parse
return
Feb  7 17:11:27 myname high2low[2605]: Status is 0
Feb  7 17:11:27 myname high2low[2605]: 74 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  7 17:11:27 myname high2low[2605]: DEBUG -- size_diff is 23
Feb  7 17:11:27 myname high2low[2605]: DEBUG -- change_made is 1
Feb  7 17:11:27 myname high2low[2605]: DEBUG -- SendPacket(): pktlen is
odd.
Feb  7 17:11:27 myname high2low[2605]: DEBUG -- SendPacket(): writing 111
bytes.
Feb  7 17:11:27 myname low2high[2607]: Entering UPD Case:
Feb  7 17:11:27 myname low2high[2607]: Starting SNMP_PARSE: Version
Feb  7 17:11:27 myname low2high[2607]: Starting SNMP_PARSE: Community
String
Feb  7 17:11:27 myname low2high[2607]: Starting SNMP_PARSE: GetResponse
Feb  7 17:11:27 myname low2high[2607]: Entering STATUS Check on parse
return
Feb  7 17:11:27 myname low2high[2607]: 97 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0
Feb  7 17:11:29 myname high2low[2605]: Entering UDP Case:
Feb  7 17:11:29 myname high2low[2605]: DEBUG - Overall Packet is
[302702010104067075626c6963a01a02024dea020100020100300e300c06082b06010201
01040005006]
Feb  7 17:11:29 myname high2low[2605]: Starting SNMP_PARSE: Version
Feb  7 17:11:29 myname high2low[2605]: Starting SNMP_PARSE: Community
String
Feb  7 17:11:29 myname high2low[2605]: Starting SNMP_PARSE: GetRequest
Feb  7 17:11:29 myname high2low[2605]: DEBUG - Entering snmp_pdu_parse()
Feb  7 17:11:29 myname high2low[2605]: DEBUG - Entering
snmp_varbind_parse()
Feb  7 17:11:29 myname high2low[2605]: DEBUG - count is [10]
Feb  7 17:11:29 myname high2low[2605]: DEBUG - len_change is [0]
Feb  7 17:11:29 myname high2low[2605]: Error is 0, ind is 1, and
elem.type is 1
Feb  7 17:11:29 myname high2low[2605]: DEBUG -- elem.data.raw is NULL
Feb  7 17:11:29 myname high2low[2605]: DEBUG -old_size is 26 (0x1A),
new_size is 26 (0x1A)
Feb  7 17:11:29 myname high2low[2605]: DEBUG -old_size is 39 (0x27),
new_size is 39 (0x27)
Feb  7 17:11:29 myname high2low[2605]: DEBUG - Overall NEW Packet is
[302702010104067075626c6963a01a02024dea020100020100300e300c06082b06010201
01040005006]
Feb  7 17:11:29 myname high2low[2605]: Entering STATUS Check on parse
return
Feb  7 17:11:29 myname high2low[2605]: Status is 0
Feb  7 17:11:29 myname high2low[2605]: 69 bytes from 192.168.1.1 to
192.168.3.2 on ep1 status = allowed 0
Feb  7 17:11:29 myname high2low[2605]: DEBUG -- size_diff is 0
Feb  7 17:11:29 myname high2low[2605]: DEBUG -- change_made is 0
Feb  7 17:11:29 myname high2low[2605]: DEBUG -- SendPacket(): writing 83
bytes.
Feb  7 17:11:29 myname low2high[2607]: Entering UPD Case:
Feb  7 17:11:29 myname low2high[2607]: Starting SNMP_PARSE: Version
Feb  7 17:11:29 myname low2high[2607]: Starting SNMP_PARSE: Community
String
Feb  7 17:11:29 myname low2high[2607]: Starting SNMP_PARSE: GetResponse
Feb  7 17:11:29 myname low2high[2607]: Entering STATUS Check on parse
return
Feb  7 17:11:29 myname low2high[2607]: 97 bytes from 192.168.3.2 to
192.168.3.1 on ep0 status = allowed 0
```

Appendix D: Frame Report from Exodus Penetration Test.

The following is an unmodified copy of the actual test results of a penetration test that was performed by Exodus personnel. This is not an interpreted report but is the actual output of the test. The output is broken down into three sections, configuration, domain exploits, and report.

Table of Contents

Frame report for netmon (netmon.com)

Hosts scanned
  Host   : 192.168.4.2 (192.168.4.2) -->

Tools used
  axfr (tcp_0_unknown)
  domainscan (all_0_unknown)
  ftpchk.gHost (tcp_21_ftp)
  http.cjr (tcp_80_http)
  httpcjrwrap (tcp_443_https)
  httpdtype (tcp_80_http)
  mail.gHost (tcp_110_pop3)
  namedscan (tcp_53_domain)
  namedscan (udp_53_domain)
  zcat (none_0_unknown)
  zcat (none_0_unknown)
  showmount (tcp_2049_shilp)
  smtp.cjr (tcp_25_smtp)
  ts (tcp_1080_socks)
  whiskerwrap (tcp_80_http)
  sslwrap (tcp_443_https)
  whois (none_0_unknown)

DANGER

```
> axfr -D /tools/home/netmon/frame_netmon_09.28.2001_11.32.01 netmon.com

Note: no 'signal' defined for this tool

Tool Name  : axfr
Description: Zone transfer agent for DNS
Protocol   : tcp
Port       : 0
Source     : axfr.def
Risk       : MEDIUM

Host    : 192.168.4.2 (192.168.4.2) -->
```

ExecResults for 'axfr -D /tools/home/netmon/frame_netmon_09.28.2001_11.32.01 netmon.com'

Returncode: 0

DANGER
> zcat /tools/home/netmon/frame_netmon_09.28.2001_11.32.01/com/netmon.com/zone*

Note: no 'signal' defined for this tool

```
Tool Name  : zcat
Description: Zcat the results of axfr so they appear in the report
Protocol   : none
Port       : 0
Source     : repaxfr2.def
Risk       : MEDIUM

Host     : 192.168.4.2 (192.168.4.2) -->
```

ExecResults for 'zcat /tools/home/netmon/frame_netmon_09.28.2001_11.32.01/com/netmon.com/zone*'

Returncode: 1

DANGER
> zcat /tools/home/netmon/frame_netmon_09.28.2001_11.32.01/com/netmon.com/hosts*

Note: no 'signal' defined for this tool

```
Tool Name  : zcat
Description: Zcat the results of axfr so they appear in the report
Protocol   : none
Port       : 0
Source     : repaxfr.def
Risk       : MEDIUM

Host     : 192.168.4.2 (192.168.4.2) -->
```

ExecResults for 'zcat /tools/home/netmon/frame_netmon_09.28.2001_11.32.01/com/netmon.com/hosts*'

Returncode: 1

DANGER
> whois netmon.com

Note: no 'signal' defined for this tool

```
Tool Name  : whois
Description: whois check of Internic info
Protocol   : none
Port       : 0
Source     : whois-1.def
Risk       : MEDIUM

Host     : 192.168.4.2 (192.168.4.2) -->
```

```
ExecResults for 'whois netmon.com'

Returncode: 1
Bad return code (expected 0)
```

```
> nmap -oM - -sS -O -P0 -n  -v 192.168.4.2

Host     : 192.168.4.2 (192.168.4.2) -->
Found open TCP ports:

 - Raw output -

ExecResults for 'nmap -oM - -sS -O -P0 -n  -v 192.168.4.2'

Output...
#

Command timed out after 900 seconds
Returncode: 0
```

```
> nmap -oM - -sU -P0 -n  -
p7,9,13,19,37,53,67,68,69,70,88,110,111,123,137,138,139,143,161,162,177,179,194,220,369,512,513,514,5
17,518,520,525,2049,6000-6010 -v 192.168.4.2

Host     : 192.168.4.2 (192.168.4.2) -->
Found open UDP ports:

 - Raw output -

ExecResults for 'nmap -oM - -sU -P0 -n  -
p7,9,13,19,37,53,67,68,69,70,88,110,111,123,137,138,139,143,161,162,177,179,194,220,369,512,513,514,5
17,518,520,525,2049,6000-6010 -v 192.168.4.2'

Output...
# nmap (V. 2.54BETA22) scan initiated Fri Sep 28 11:32:17 2001 as: /usr/bin/nmap -oM - -sU -P0 -n -
p7,9,13,19,37,53,67,68,69,70,88,110,111,123,137,138,139,143,161,162,177,179,194,220,369,512,513,514,5
17,518,520,525,2049,6000-6010 -v 192.168.4.2
# Ports scanned: TCP(0;) UDP(44;7,9,13,19,37,53,67-70,88,110-111,123,137-139,143,161-
162,177,179,194,220,369,512-514,517-518,520,525,2049,6000-6010)
Host: 192.168.4.2 ()     Status: Up
# Nmap run completed at Fri Sep 28 11:33:17 2001 -- 1 IP address (1 host up) scanned in 60 seconds

Returncode: 0
```
